NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A STATISTICAL ANALYSIS AND ASSESSMENT OF THE IMSI-CATCHING THREAT AGAINST MOBILE SECURITY STANDARDS

by

Carmen A. Johnson

June 2020

Thesis Advisor:	Chad A. Bollmann
Co-Advisor:	Carson C. McAbee
Second Reader:	John D. Roth

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2020	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE A STATISTICAL ANALYSIS AND ASSESSMENT OF THE IMSI-CATCHING THREAT AGAINST MOBILE SECURITY STANDARDS	5. FUNDING NUMBERS
6. AUTHOR(S) Carmen A. Johnson	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

International mobile subscriber identity (IMSI) catching is a man-in-the-middle attack that utilizes rogue base stations to intercept the IMSIs of mobile users. Attackers can use software-defined radios (SDR) and open source software to create rogue base stations that geolocate or execute other malicious attacks against their targets. Prior work proves that attackers are not limited to targeting either old or new cellular devices since current devices are interoperable with older mobile networks, including GSM. The goal of this thesis is to determine if cellular devices are susceptible to target profiling based on the model or manufacturer of the device. If devices can be profiled, then can attackers improve rogue base stations to capture devices faster? To answer this, we created an enclosed test network using SDRs and OpenBTS to mimic GSM base stations. We strived to eliminate the factors that devices use to select base stations. We then presented an IMSI-catching program that can configure base stations, capture IMSIs, and log base station selection data for analysis. Finally, we conducted a set of experiments to assess if cellular devices have connection preferences that can be profiled. The results of the experiments suggest that we were not able to successfully eliminate some decision-making factors. However, more rounds and an examination of the factors that could have affected the outcome are required to make any conclusions on the selections that were exhibited.

14. SUBJECT TERMS IMSI, IMSI-catching, mobile security, rogue base station, GSM	15. NUMBER OF PAGES 129
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A STATISTICAL ANALYSIS AND ASSESSMENT OF THE IMSI-CATCHING THREAT AGAINST MOBILE SECURITY STANDARDS

Carmen A. Johnson
Lieutenant, United States Navy
BS, Utica College, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2020

Approved by: Chad A. Bollmann
Advisor

Carson C. McAbee
Co-Advisor

John D. Roth
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

International mobile subscriber identity (IMSI) catching is a man-in-the-middle attack that utilizes rogue base stations to intercept the IMSIs of mobile users. Attackers can use software-defined radios (SDR) and open source software to create rogue base stations that geolocate or execute other malicious attacks against their targets. Prior work proves that attackers are not limited to targeting either old or new cellular devices since current devices are interoperable with older mobile networks, including GSM. The goal of this thesis is to determine if cellular devices are susceptible to target profiling based on the model or manufacturer of the device. If devices can be profiled, then can attackers improve rogue base stations to capture devices faster? To answer this, we created an enclosed test network using SDRs and OpenBTS to mimic GSM base stations. We strived to eliminate the factors that devices use to select base stations. We then presented an IMSI-catching program that can configure base stations, capture IMSIs, and log base station selection data for analysis. Finally, we conducted a set of experiments to assess if cellular devices have connection preferences that can be profiled. The results of the experiments suggest that we were not able to successfully eliminate some decision-making factors. However, more rounds and an examination of the factors that could have affected the outcome are required to make any conclusions on the selections that were exhibited.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3GPP	3rd Generation Partnership Project
AGCH	Access Grant Channel
ARFCN	Absolute Radio-Frequency Channel Number
AuC	Authentication Center
BCCH	Broadcast Control Channel
BSS	Base Station Subsystem
CCCH	Common Control Channel
CGI	Cell Global Identification
DF	Dedicated File
EF	Elementary File
EIR	Equipment Identity Register
FACCH	Fast Associated Control Channel
FCCH	Frequency Control Channel
GSM	Global System for Mobile Communications
HLR	Home Location Register
LA	Location Area
LAI	Location Area Identity
LAC	Location Area Code
ICCID	Integrated Circuit Card Identifier
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
MCC	Mobile Country Code
MF	Master File
MM	Mobility Management
MNC	Mobile Network Code
MS	Mobile Station
MSIN	Mobile Subscriber Identification Number
NSS	Network and Switching Subsystem
PCH	Paging Channel
PLMN	Public Land Mobile Network

RF	Radio Frequency
SACCH	Slow-Associated Control Channel
SCH	Synchronization Channel
SDCCH	Stand-alone Dedicated Control Channel
SDR	Software-Defined Radio
SIM	Subscriber Identification Module
TCH	Traffic Channel
UE	User Equipment
VLR	Visitor Location Register

ACKNOWLEDGMENTS

I would first like to thank my advisors, CDR Chad Bollmann and Carson McAbee, for their time, guidance, and patience during this process. Their immense knowledge and dedication have made this work possible, and I am truly thankful to have worked with them.

I would also like to thank Bob Broadston for all of his assistance and the resources spent in troubleshooting test equipment.

Finally, I would like to sincerely thank my parents for their continuous encouragement and support throughout my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Global System for Mobile Communications (GSM) Association (GSMA) estimated that in 2019, approximately 3.8 billion people relied on their mobile device to connect to the internet. By 2025, this figure is expected to increase to 5 billion [1]. The number of internet of things (IoT) devices, including cellular-based IoT, is growing even faster and has already reached 12 billion. This figure is expected to grow to 24.6 billion by 2025 [1]. IoT devices typically connect devices such as cars, smart grid components, and appliances to the internet. Some IoT devices use a cellular network for this connection [2]. The latest smartphones and IoT devices using cellular networks inherit vulnerabilities when maintaining backward compatibility with older cellular network standards [3]. One of these vulnerabilities includes a susceptibility to privacy attacks using international mobile subscriber identity (IMSI) catching.

IMSI catching is a man-in-the-middle attack that uses rogue base stations, or IMSI catchers, to capture subscriber information in order to track or to further other malicious motivations [3]. The IMSI catching attack was first developed to exploit the fact that cell phones do not authenticate GSM networks, allowing rogue base stations to replicate actual GSM base stations [3].

As of 2019, GSM networks are not as widely used as 4G, with 4G making up approximately 52% of connections globally, not including the cellular based IoT devices [1]. Research has identified methods to perform attacks similar to IMSI catching even on newer networks [2]. Additionally, attackers are able to downgrade even newer devices to GSM for ease of capture [4].

From a cyber security point of view, IMSI catching poses a threat to consumers worldwide using either old or new technologies. This threat becomes especially worrisome in environments with low cell coverage as there is a higher likelihood of success for rogue base stations to provide the best signal service in the area. However, attackers are not just limited to low signal areas to provide the best cell signal. Attackers can also actively jam

the surrounding legitimate options or leverage environmental factors to passively jam any authentic options [3].

A. THESIS OBJECTIVE

The main goal of this thesis is to explore a possible method of targeting IMSI catching by determining if cell phones and IoT devices can be profiled by attackers. In order to examine this possibility, the objectives of this thesis are to:

1. Investigate the components within a GSM network to determine the factors that contribute to a device's decision-making process for selecting a cell tower

2. Build a closed test environment that mimics a GSM network

3. Build a set of IMSI catchers that are reconfigurable and are able to collect and record cellular device connection preferences for statistical analysis

4. Implement a set of experiments examining the individual factors that contribute to cell selection

The goal of the analysis work is to determine the potential for profiling mobile devices based on their base station selections during experimentation. If profiling can be accomplished, then this indicates that attackers have the ability to capture IMSIs quicker when knowing the model of the target device. The work in this thesis provides the design of a testing environment along with the programs used to catch IMSIs and measure selection characteristics for analysis.

B. RELATED WORK

The work in this thesis aims to implement the characteristics of an IMSI catcher using OpenBTS [5] and software-defined radios (SDR) in a manner similar to the prior work discussed in this section. The 3rd Generation Partnership Project (3GPP) specifications [6]–[12] are the technical standards that provide the information necessary to set up a GSM test network and implement a set of IMSI catchers.

Several research works have examined the topic of IMSI catching. Many of the GSM attacks and all of the open source software available that makes IMSI catching both affordable and easily implemented on the GSM network are discussed in [13]. The work in [14] presented an analysis of the GSM traffic produced through OpenBTS and an SDR operating as an IMSI catcher. Strobel [15] discussed the different approaches to IMSI catching attacks on a GSM network and an approach for doing so on a Universal Mobile Telecommunications System (UMTS) network.

The works of Retterstol [4], Mruz [16], and Debrowski et. al. [17] all used a similar OpenBTS and SDR test platform to confirm that IMSI catching could be performed on devices using newer cellular standards. The works confirm the interoperability of networks between phone models. Retterstol [4] presents an implementation of an IMSI catcher that improves on the work of IMSI catching research predecessors. Retterstol [4] and Weinmann [3] also confirm more advanced malicious attacks that could be further implemented after or while catching an IMSI. Debrowski et. al. [17] and Mruz [16] broaden the scope of previous work to present an analysis of IMSI catching and methods to detect IMSI catchers.

C. ORGANIZATION

This thesis is split into five chapters along with seven appendix sections. A moderate review of the GSM architecture is performed in Chapter II, covering the functionality of everything from a subscriber identification module (SIM) card to the external networks of a local GSM network. Chapter III discusses the structure of the test environment as well as the investigative efforts that helped form the experiments. Chapter IV presents the results of the experiments. The conclusions made from the experiments and any future work that can stem from this thesis are discussed in Chapter V. Finally, the code used for the test environment and experimentations is revealed in the appendixes.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The latest generations of smart phones and devices continue to possess the GSM-era vulnerabilities due to their backward compatibility with the GSM technology. This chapter explores the fundamentals of GSM, its components, and the interactions that take place between those components. Within the discussion of the GSM components are the inner workings of a cell phone and a review of the architecture from the cell tower to the switching stations within a mobile network. The radio frequency (RF) resources and communication procedures required for the cell phone and cell tower to successfully communicate are also discussed. The chapter then concludes with an introduction to rogue base stations.

A. MOBILE STATION

The cell phone, referred to as a mobile station (MS), is composed of two parts. The first portion is called the user equipment (UE) in the 3GPP standards but is also commonly referred to as mobile equipment (ME). The UE works to meet the demands of the user through both the application and radio frequency (RF) interfaces. The second portion is the SIM card, housing important subscriber data and information that assists in initializing and maintaining communications [18]. The aforementioned sections work together to enable users to communicate within a mobile network.

1. User Equipment

The UE is identified by a globally-unique International Mobile Equipment Identifier (IMEI) and contains two major processors, an application processor and a modem [19]. The modem is frequently referred to as the baseband processor and the IMEI is more accurately the identifier for the modem within the UE [10]. However, it is quite commonly stated that the IMEI identifies the UE. The decision-making processes for MSs rely on these processors for maintaining connectivity and communications to a mobile network.

a. Application Processor and Modem

Most modern smart phones contain two major processors mentioned previously as the application processor and the modem. The application processor handles the processes that have really given smart phones their “smart” label, such as the Android or iOS operating systems, the underlying Linux kernel, and all the applications downloaded from app stores [3]. However, once the user requires a cellular action, such as sending a text message or making a phone call, the responsibility for that action falls on the modem [19].

The modem has the major responsibility of acquiring and maintaining a connection to a mobile network in order to perform cellular actions at a moment’s notice. It does this through three major sections of the modem: 1) the RF frontend, 2) the analog baseband section, and 3) the digital baseband section [19]. The RF frontend is shown in the blue section of Figure 1 and interconnects the MS with the air interface, which is a medium discussed in Section C. The analog baseband section is the orange section consisting of the integrated chip combinations necessary to perform two major functions. The first is to demodulate signals from the cell tower and send them to the digital baseband section. The second function is to convert the digital signals from the digital baseband section into an analog signal ready for the RF frontend to emit via the air interface [19]. The digital baseband system, highlighted in green in Figure 1, facilitates the communication between the application processor and the cellular network.

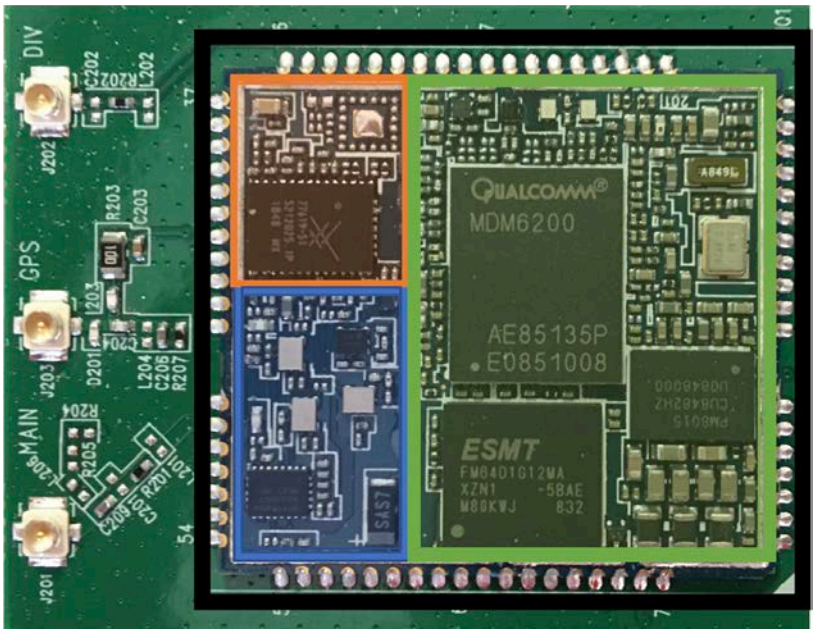

Figure 1. Inner Workings of a Quectel UC20 Modem

b. Communication between Processors

Smart phones with both application and baseband processors require a method for communication to flow between the two processors. For example, when a user uses the phone application on a smart phone to make a phone call, the application processor must somehow inform the baseband processor that a phone call needs to occur. This communicative process occurs by one of two layouts: either by having memory space shared between the two processors, or through using a serial link between processors, as shown in Figure 2 and described in [3], [19]. Much of the information about the architecture and specific communication protocols between the two processors is proprietary knowledge of the processor manufacturers. Most information that has been found is from thorough analysis work by security professionals and not provided by the manufacturers themselves [20].

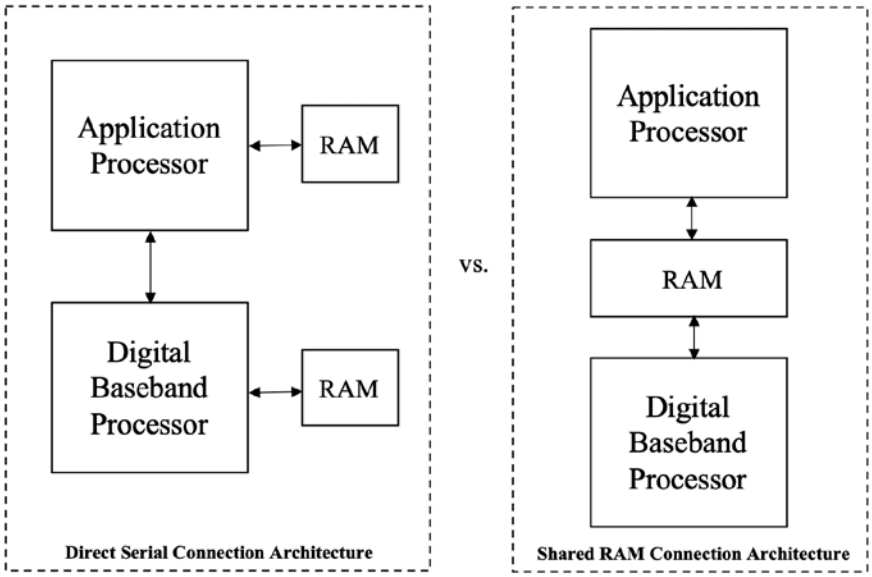

Figure 2. Processor Communication Architectures. Adapted from [3].

While the procedures for the processors are proprietary knowledge, there are standardized communication commands, known as attention (AT) commands, that are well known. The application processor uses AT commands as a way to request a cellular action. AT commands can also be used to request specific subscriber information from the SIM card. A considerably large number of AT commands have been approved under 3GPP TS 27.007 [10], but it is worth noting that each modem may differ in its ability to carry out a specific AT command. Cellular modem manufacturers may have released manuals describing what specific AT commands are permitted and their usage for a particular modem model. This is the case for the UC20 modem displayed in Figure 1 [21].

The core processor within the modem processes and handles the functional AT commands [19]. Using Figure 1 as an example, the core processor is identifiable as the larger chip within the green section, the Qualcomm MDM6200. More recent modems, such as the Quectel EC20, have cores that run a Linux distribution from the OpenEmbedded [22] framework shown in [20]. This Linux distribution is designed specifically to run on an embedded device and is used to process the AT commands given to the modem, or more specifically the digital baseband processor within the modem, as described in [19].

While most modems have a fairly standard layout, the ways in which all of these processors communicate with one another on a deeper level is not as well known. To further understand how the MS works together to perform basic cellular activities, it is important to also analyze the second component to the MS. Several situations exist where the modem may require information that is no longer stored on the UE, but instead stored on the SIM card.

2. SIM Card

Aside from the UE within the MS there is the SIM card, uniquely identified by an integrated circuit card identifier (ICCID) defined by ITU E.118 [23]. The SIM card itself may vary in size; however, it holds all of the subscriber's information that is necessary for authentication and communication on a cellular network. Such information includes an IMSI, a temporary mobile subscriber identifier (TMSI), broadcast control channel (BCCH) information, a location area identity (LAI), forbidden public land mobile networks (FPLMN), preferred public land mobile networks (PLMN), an authentication key (K_i), etc. [24]. The vast information within the SIM card has a layered structure to support the necessary functions carried out by the modem.

a. Important Elements Stored on a SIM Card

One of the most important elements in the SIM card is the IMSI, which is used to identify a subscriber on a network. Within the GSM network the MS uses the IMSI to initiate a desire for connection to a network [24]. The IMSI is a 15-digit identifier that is composed of a three-digit mobile country code (MCC), two-digit mobile network code (MNC), and a ten-digit mobile subscriber identification number (MSIN) [25]. As the names suggest, the MCC identifies the country and the MNC identifies the network within that country. The MSIN indicates the subscriber under their home network defined by the MCC and MNC. Similarly, the network can temporarily assign a TMSI to be used in place of the IMSI. This was implemented as a way to provide some form of anonymity, in hopes that eavesdroppers cannot map the TMSI to the IMSI of the subscriber [25]. The K_i is a unique private key used to authenticate the subscriber on a network, a process that takes place after the IMSI has already been used to identify the user to a network [24]. Much of the work in

later chapters deals strictly with the processes that take place leading up to the IMSI being sent to the network for access to communications; therefore, authentication will not be discussed further here. However, the information regarding the BCCH information, LAI, and PLMN's is explained in more detail in sections C.3, B.1, and B.3, respectively, of this chapter.

b. Organizational Structure of the SIM Card

The SIM card houses a file system, defined by GSM 51.011 [12], that is organized in a hierarchical structure with four main levels, shown in Figure 3. Each level can be identified by a two-byte file identifier. The master file (MF) on the top level holds the file identifier 3F*XX* and is essentially the root directory of all the files in the SIM card. The first level under the MF are dedicated files (DF) with the file identifier 7F*XX*, or 2F*XX* for elementary files (EF) on this level. The DFs are considered to be directories rather than files themselves as they are essentially the folders for the EFs or other DFs. The second level under the MF can also contain DFs and EFs, with file identifiers 5F*XX* and 6F*XX*, respectively. The third level will only contain EFs that are under their corresponding second level DF as 4F*XX*. The individual locations for information stored under an EF are known as binaries, or transparent files, and are displayed as an arrangement of bytes [12]. When the modem is attempting to acquire information from a SIM card, it will be reading the binary of an EF for that information. Likewise, if the modem needs to update information to the SIM card it will update the bytes of the EF binary.

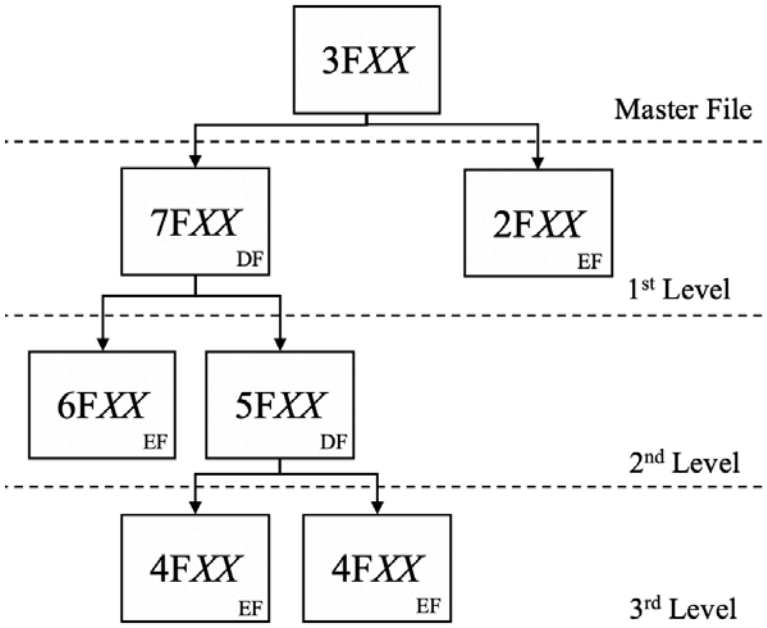

Figure 3. SIM Card Filesystem. Adapted from [12].

The amount of information stored, along with their respective locations, is overwhelmingly large. Much of the information extends past the requirements for understanding the work of this thesis. However, the DF_{GSM}–7F2*X* is the container for most of the important location and subscriber information for communicating on a GSM network [12]. The EFs that are of focus within DF_{GSM} are: 6F07–$EF_{IMSI,}$ 6F74–EF_{BCCH}, 6F7E–EF_{LOCI}, 6F7B–$EF_{FPLMN,}$ and 6F30–$EF_{PLMNsel}$.

The address space for EF_{IMSI} simply stores the IMSI for the subscriber to use on a network, as defined by [12]. The 6F74 file identifier for EF_{BCCH}, stores the neighbor cell information from system information two messages for both cell selection and reselection procedures [12]. Cell selection and reselection are discussed in more detail within Section C of this chapter.

The location information EF, as defined in [12], stores four main elements of information within its address space: 1) TMSI, 2) LAI, 3) TMSI time, and 4) location update status. The TMSI had been briefly mentioned in Section A.2.a of this chapter and the LAI will be discussed in more detail in Section C. The TMSI time is a timer value

that is determined by the currently connected network. The time value counts down for the periodicity a MS should provide a location update to the network, specified in [9]. The location update status indicates whether or not the last location update attempt was updated successfully or attempted to provide a location update to a forbidden network [9].

The service provider determines the addresses in the space dedicated for storing a list of FPLMNs and each FPLMN are three bytes in length [12], [26]. The provided space for FPLMNs are defined to be 12 bytes long, which means only four FPLMNs can be stored on the SIM card. Lastly, the 6F30 file identifier for $EF_{PLMNsel}$ allows for storing a list of preferred PLMNs in order of priority which a MS should use to search for [12]. The details of PLMNs are discussed in more detail in Section B.

The UE and SIM card together form the MS that communicates on a cellular network and the UE uses a modem as the primary processer for doing so. The SIM card contains information stored in a hierarchical structure for the modems use in communicating with a GSM network.

B. NETWORK ARCHITECTURE

The physical network beyond the MS itself consists of many components and systems that can fundamentally be divided into three main subsystems, as shown in Figure 4. The base station subsystem (BSS) manages the radio interface infrastructure, providing the resources for communication flow to and from the MS. The network and switching subsystem (NSS) connects the BSS to the external network and handles the user data traffic flow [24]. The final subsystem is the Operation and Maintenance Center (OMC), which will not be discussed at length since it is outside the scope of this thesis. The OMC interfaces with the subsystems of the GSM architecture for network maintenance and subscription management. Together, the subsystems work to enable communication flow between users [18].

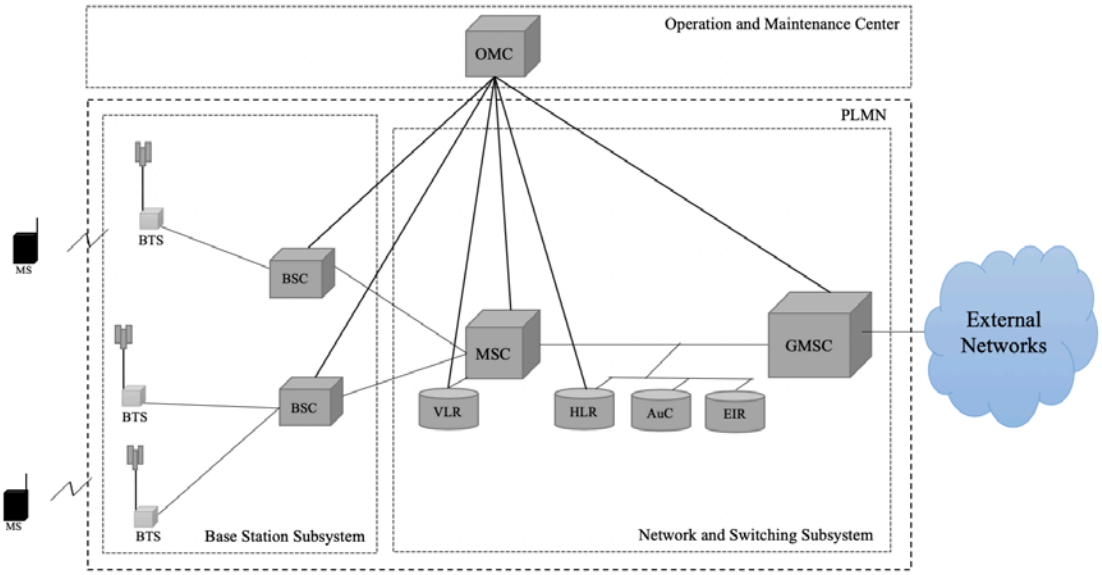

Figure 4. GSM Architecture. Adapted from [18].

1. Base Station Subsystem

As briefly mentioned earlier, the BSS provides the necessary radio infrastructure for a location area (LA) and is broken down into two main stations. The first element is the base transceiver station (BTS) and the second element is the base station controller (BSC) overseeing the BTSs within a LA [25].

The BTS is commonly recognized from their antennae as the cell tower and is the first communication element for the MS on the GSM network. Geographic locations of a network are divided into cells, each containing a BTS [24]. The BTS is in charge of the signal processing that is required for relaying uplink and downlink communication to the MS's within its cell. The size of each cell is not standardized, since things like population density and geographic features of the location are some of the many factors in determining cell tower locations and their individual breadth of responsibility [24].

The BSC controls and manages a group of BTSs for a LA and is identified by a LAI. The LAI itself is comprised of the MCC, MNC, and location area code (LAC), defined in [8]. The BSC primarily relays communications and to carry out that main objective, the BSC must manage and maintain connectivity to MSs [24]. For example, the BSC handles

the connectivity transition, or handoff, between cells when a MS is moving from cell to cell within a LA. The BTSs directly interface with the MSs on the air interface via physical channels and logical channels. The BSC is overall in charge of the management and allocation of those physical and logical channels as well as the traffic sent to and received from the next subsystem [24].

2. Network and Switching Subsystem

The NSS handles the switching of the user data traffic flow throughout its territory by using many important elements. The first element is the mobile switching center (MSC) controlling the traffic flow to the appropriate LAs. The other four elements are one authentication center (AuC) and three registers. Those three registers are known as the equipment identity register (EIR), home location register (HLR), and visitor location register (VLR). The AuC and registers store information or perform a specific function for the MSC on the GSM network [18].

a. Mobile Switching Center

The MSC handles the large amount of user data traffic, but additionally has to work together with the BSS to handle radio resource allocation to the MSs [24]. There is a difference between maintaining radio resource allocation at the BSC level versus the MSC level. Recall the scenario described for the BSC maintaining communication for an MS as it moves from cell to cell in a LA. The radio resource allocation and management at the MSC level occurs when a MS is moving from a cell that is in one LA to an adjoining cell that is in a different LA. The MSC oversees a group of BSCs, each in charge of their respective LAs. Frequently, a Gateway MSC (GMSC) is used to group MSCs and provide the bridge to the external networks. The GMSC and MSC level also maintain specific databases used to perform the other necessities for a cellular network [24].

b. Databases

As mentioned earlier, aside from the MSC there are four other elements within the NSS: AuC, EIR, HLR, and VLR [24]. The AuC has the sole purpose of authenticating subscribers to a network. The UE itself does not technically go through an authentication

procedure like the SIM card does with the AuC. However, the EIR contains lists of IMEIs to determine authorization of UEs on a network [24]. This is helpful in situations such as the case of a stolen phone. The HLR and VLRs have a similar relationship with each other since subscriber information such as the IMSI, locational data, and phone number are stored in both the HLRs and the VLRs. The major difference between the two registers is that there is one HLR for a network while there is typically a VLR for every MSC in the GSM network. The VLR dynamically changes as MSs move throughout its purview. For instance, when a MS attaches to a network in a new LA, the VLR servicing the MSC in that LA will add the subscriber and its location to its records. The VLR as well can assign a TMSI, for the MS to use instead of using an IMSI as a form of identification while it is in the VLR's LA [24].

3. Public Land Mobile Network

A PLMN encapsulates a GMSC and all of its attached subsystems, also shown in Figure 4. The PLMN is identified by two elements mentioned before as the MCC and MNC, or the country and region that defines the borders of that PLMN [25]. A single PLMN is essentially a container for a group of MSCs belonging to a network provider within a defined geographic region. The PLMN technically is not a physical component on a network, but rather a collection of components that organizes and sets boundaries for users to access cellular services under a telecommunication provider [24].

C. UM INTERFACE

The first sections discussed most of the physical elements of a GSM network without clarifying how the network itself is connected to the MS. The Um interface links the MS and the BTS, with more common references as the air, radio, or RF interfaces. This section first provides an introduction to the physical and logical layers of the air interface, and then discusses the elements that allow the physical and logical layers to work. The layers of the air interface allow for the transmission of data, which includes the data for cell selection and reselection. Additionally, the layers of this air interface work together to create a functional link between the MS and BTS in order to enable and maintain communications, tying the MS to the GSM network [24].

1. Physical and Logical Channels

As previously mentioned, geographic regions of a network are further divided into cells that ensure communication coverage to the bounds of each cell. Furthermore, those cells enable communication to take place using portions of the RF spectrum. For GSM, the cells are limited to operating within four main bands and their dedicated absolute radio frequency channel number (ARFCN) ranges, as shown in Table 1. ARFCNs will be discussed further in Section b, however, there are other outdated bands or extensions that are not shown. For relevancy to later chapters of this thesis and simplicity, the discussion will focus on the bands within Table 1. The air interface for those bands takes advantage of both frequency division multiple access (FDMA) and time division multiple access (TDMA) [24]. Combining these access methods enables multiple users to communicate at the same time, and to efficiently use a limited range of bandwidth.

Table 1. GSM Bands and Channels. Adapted from [24].

Band	Uplink Range (MHz)	Downlink Range (MHz)	ARFCNs
850	824–849	869–894	128–251
900	890–915	935–960	1–124
1800	1710–1785	1805–1880	512–885
1900	1850–1910	1930–1990	512–810

a. Physical Layer

The physical layer is formed by first dividing the available air interface into the uplink and downlink ranges, provided in Table 1. Uplink refers to the communications that are going from the MS to the BTS. The downlink is for communications from the BTS to the MS [24]. For the case of the 900 band, shown in Figure 5, there is a 45 MHz offset between the two ranges. The offset is used to avoid any interference of communications between the two physical ranges. The physical channels, also referred to as carrier frequencies, are

created by using FDMA to further divide the downlink and uplink ranges into channels of 200 kHz bandwidth each [24].

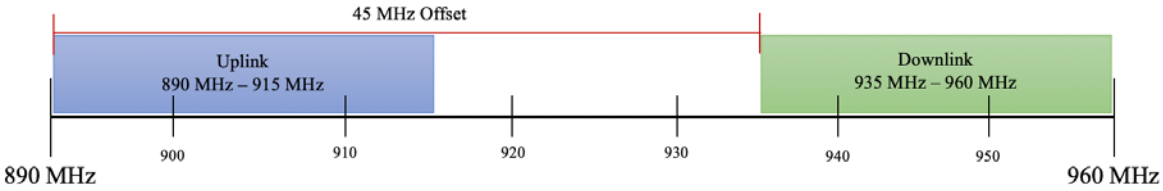

Figure 5. Uplink and Downlink Spectrums for Band 900

b. Logical Layer

Sitting on top of a physical channel is a TDMA frame containing eight timeslots (0–7) that are each allotted 576.9 µs of transmission time, shown in Figure 6. The TDMA frame must go through a constant rotation of its eight time slots within its physical channel, since time slots will only be able to transmit data on each of their individual turns [24]. The eight time slots together have an overall period of approximately 4.62 ms and each time slot will have to wait 4.038 ms before its next turn. The logical layer then operates on top of the physical layer through the use of logical channels and bursts [18].

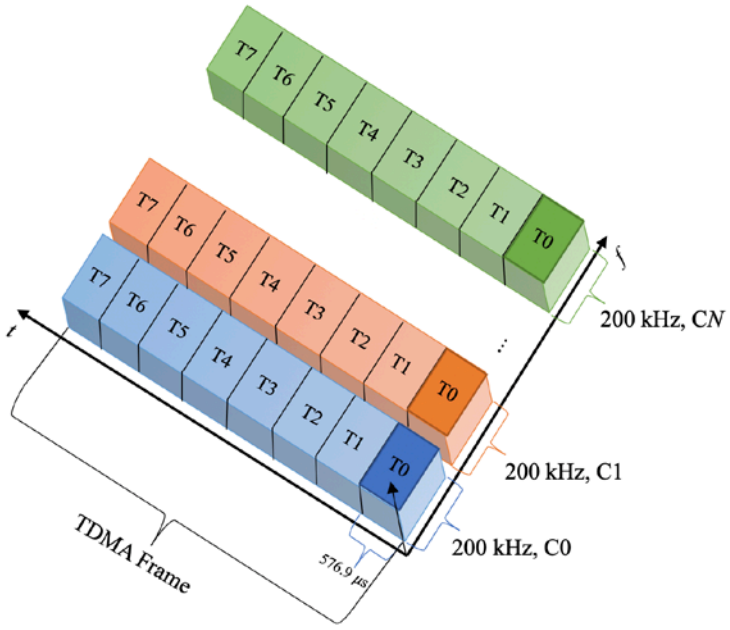

Figure 6. Physical and Logical Channels. Adapted from [18].

(1) Logical Channels

A type of logical channel will be used during a timeslot to send information to a specific target. As made apparent from Table 2, a variety of logical channels are organized into main two groups [18]. One group consists of traffic channels (TCH) which are used for transmitting data and speech at half or full rates. While a full-rate TCH supports only one user at a time, a half-rate TCH may be shared by two users. The other group contains control channels (CCH), which are also referred to as signaling channels. The CCHs are used to transmit the signaling and synchronization data necessary for establishing and maintaining communications with local MSs. The CCHs are further subdivided into three types: broadcast channels (BCH), common control channels (CCCH), and dedicated control channels (DCCH) [18].

The BCH channel type is broadcast by the BTS on the downlink in order to provide the network information MSs need to make decisions for cell selection/reselection. The BCH channels consist of broadcast control channels (BCCH), frequency correction channels (FCCH), and synchronization channels (SCH) [18]. The CCCHs are used to provide an avenue of communication between the BTSs and the MSs that are not assigned

a dedicated channel through the use of random-access channels (RACH), access grant channels (AGCH), paging channels (PCH), and notification channels (NCH). Finally, there are three various types of DCCHs, known as stand-alone dedicated control channels (SDCCH), slow associated control channels (SACCH), and fast associated control channels (FACCH) [18].

Table 2. Logical Channels. Adapted from [18].

Logical Channels											
Traffic Channels (TCH)		Signaling/Control Channels (CCH)									
Full Rate Traffic Channel (TCH/F)	Half Rate Traffic Channel (TCH/H)	Broadcast Channels (BCH)			Common Control Channels (CCCH)				Dedicated Control Channels (DCCH)		
		Broadcast Control Channel (BCCH)	Frequency Correction Channel (FCCH)	Synchronization Channel (SCH)	Random Access Channel (RACH)	Access Grant Channel (AGCH)	Paging Channel (PCH)	Notification Channel (NCH)	Stand-alone Dedicated Control Channel (SDCCH)	Slow Associated Control Channel (SACCH)	Fast Associated Control Channel (FACCH)
DL/UL	DL/UL	DL	DL	DL	UL	DL	DL	DL	DL/UL	DL/UL	DL/UL

DL: Downlink Only

UL: Uplink Only

DL/UL: Both Uplink and Downlink

Many of the time slots of the physical channels are assigned to dedicated traffic channels in order to provide and maintain lines of communications to authorized MSs in a BTS cell. Accordingly, in the case of dedicated channels for a given user, there is a downlink timeslot on a downlink physical channel and an associated uplink timeslot on an uplink physical channel [24]. For the uplink and downlink pair, the uplink timeslot is delayed from the associated downlink time slot by a span of three timeslots. The delay is required in order to avoid having the MS transmit and receive at the same. The physical channel pairing available for use as dedicated user channels are known as ARFCNs. If the

ARFCN is known, the carrier frequency for the physical channel on the uplink and downlink can both be calculated using

$$f_{900,uplink} = 890 + .2 * ARFCN$$

$$f_{900,downlink} = f_{900,uplink} + 45,$$

with the 900 band again as the example, given in [24]. Likewise, for the 900 band the ARFCN can be calculated using

$$ARFCN = \frac{f_{900,uplink} - 890}{.2} = \frac{f_{900,downlink} - 45 - 890}{.2},$$

when the frequencies of either the uplink or downlink are known.

(2) Bursts

When data is being sent during a time slot by a logical channel, the data is sent in the form of bursts [18]. A burst has a length of 156.25 bits, including the duration of bits dedicated for the guard period. The bursts can take the form of one of five standard formats shown in Figure 7. The normal burst is used to transmit speech data or signaling information depending on the channel type that uses it. Before a MS can transmit information with a BTS, it must first be able to find the BTS. The frequency correction burst works to accomplish this by using 142 of the 156.25 bits to broadcast a specific waveform. The MS uses the broadcasted waveform to tune its oscillator to the correct frequency of the BTS. As well, an MS is required to align with the time of the BTS, which is accomplished through the use of the synchronization burst. The synchronization burst provides the MS the 64-bit synchronization sequence necessary for BTS time alignment. The next burst option is the dummy burst and is only used when no data needs to be sent to fill the timeslot on the downlink. The access burst is the final burst type and is exclusively used for the RACH by the MS [18].

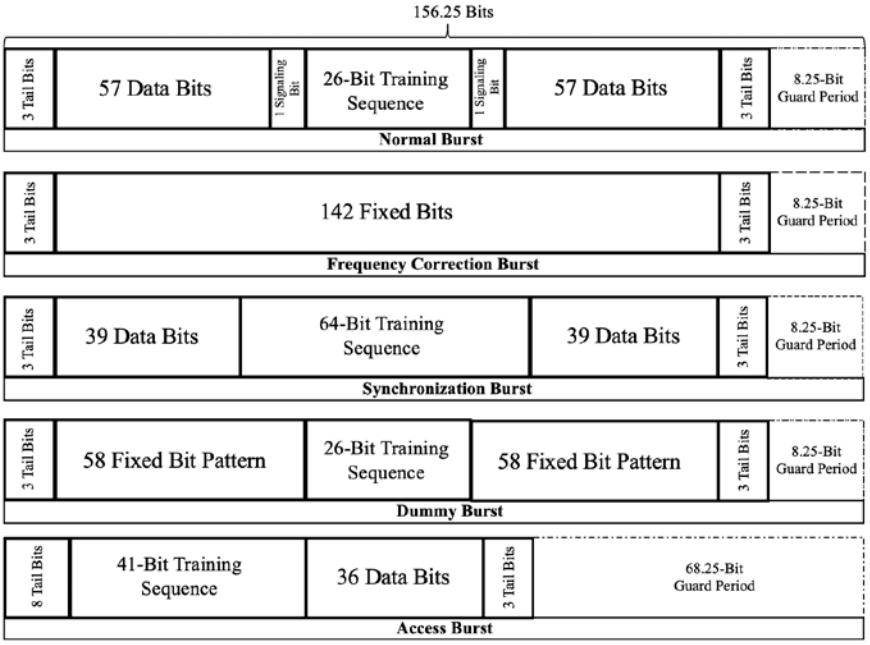

Figure 7. Burst Type Standards. Adapted from [18].

c. Channel 0 Timeslot 0

Physical and logical channel combinations existing on specific time slots generally serve a specific purpose. Physical channel 0 and timeslot 0 (C0T0) is an essential downlink channel and timeslot combination for the cell selection and reselection process which will be discussed in Section 2 of this chapter. This combination, also known as combination V, blends specific logical channel types for the purpose of enabling MSs to find and initiate a line of communication with the BTS, a vital requirement for rogue base stations. Combination V commonly takes the form, defined by GSM 5.02 [6], as: FCCH + SCH + BCCH + CCCH + 4 SDCCH + 4 SACCH. An example of the channel combination order is provided in Figure 8.

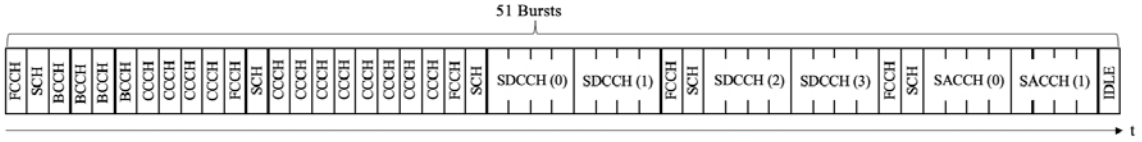

Figure 8. Combination V Example for C0T0 Downlink. Adapted from [18].

As seen in the implementation of combination V in Figure 8, the FCCH is the first channel type to broadcast data on C0T0. The FCCH uses the frequency correction burst to allow the MS to tune to the correct frequency [18]. The SCH is then used to send a synchronization burst on the next go around of C0T0s turn, to align the MS with the time of the BTS. The BCCH is next and emits four normal bursts worth of data during C0T0s next four turns. The four BCCH bursts are used to send one system information (SI) message and is done so to prevent burst errors through an interleaving process that spreads the data across the four bursts [18].

The standard provides six main SI types; however, SIs one through four are the ones broadcast regularly. The four main SIs contain important information about the network with a great deal of overlap between them [11]. For the sake of continuing the discussion on channels, more details of the SI messages will be provided as they become relevant. For now, the important detail provided within the SI messages concerns the RACH. The RACH is under the CCCH type and is also the only uplink CCCH type. The details of the RACH for communicating with a BTS are provided within one SI transmitted from the four BCCH bursts. The purpose of the RACH is to provide an avenue for the MS to request a SDCCH from the BTS [18].

Moving back to the downlink timeline, the next four bursts after the four-burst BCCH are for the CCCH. The RACH is an uplink CCCH, therefore the channels meant for the CCCH type on the downlink can only be a AGCH, PCH, or NCH [18]. AGCHs are used to assign a SDCCH to the MS. Just like the BCCH, the AGCH requires four bursts to transmit a channel assignment message to the MS containing the SDCCH details for a temporary line of communication. The PCH and NCH are only used as ways to find or notify a MS within the cell.

The next two bursts are another FCCH and SCH, followed by eight CCCH bursts before another two FCCH and SCH pair. These are extra opportunities provided to MSs to correct, synchronize, and/or get access to a dedicated channel since the next eight bursts are for two SDCCHs, four bursts for each SDCCH. These SDCCHs are again followed by another pairing of the FCCH and SCH with one more set of two SDCCHs (eight bursts). One last FCCH and SCH are followed by two SACCHs, at four bursts each, and one idle

burst to finish a 51-burst cycle. Since a time slot is required to emit a burst even if there is no data to send, the idle burst is one example that uses the dummy burst. Combination V requires support for four SACCHs; therefore, the entire 51-burst frame from Figure 8 is repeated [18]. The SACCH (0) and SACCH (1) will then be SACCH (2) and SACCH (3), respectively. The SACCHs are used to provide the MSs within the cell, signaling and channel measurements and are continuously doing so on the SACCH turns. To send one full 51-burst frame of combination V should take around .231 seconds, with 50 bursts * (576.9 μs burst period * 8 time slots) + 576.9 μs for burst 51 of C0T0. Therefore, two cycles of combination V to include the transmission of all four SACCHs, takes approximately .467 seconds.

If the BTS uses this configuration for C0T0, the MS has one opportunity to gather the four BCCH bursts with the necessary information. Otherwise, the MS will have to wait through what is remaining of the 51-burst cycle before repeating to get the next round of BCCHs. This requirement is due to the necessity of getting the RACH information, so the MS knows the avenue to request for a dedicated channel. Once the channel is given to the MS from the AGCH, the MS can then go through a location update procedure with the BTS if it selects that cell for services [24]. The location update is initialized by sending the IMSI to identify the subscriber to the servicing network, as shown in Figure 9. A discussion of procedures such as a location update and cell selection/reselection is found in Section C.2.

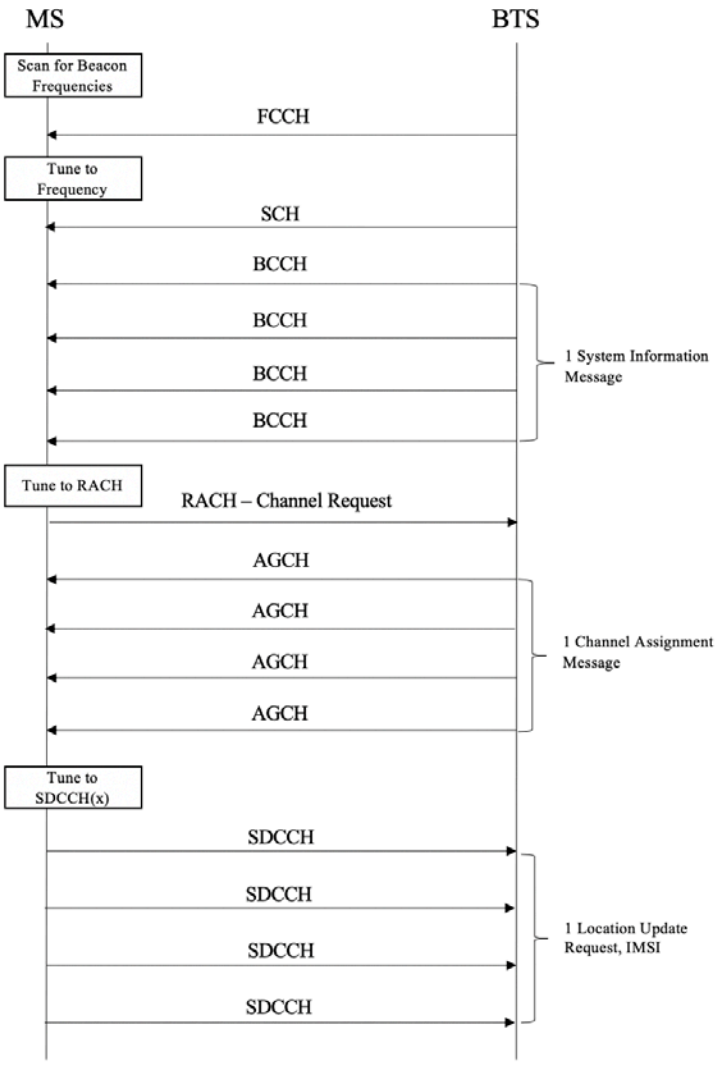

Figure 9. Simplified Process for Initializing Location Update

In summary, the physical and logical layers provide the medium and the resources necessary to communicate over the air interface. In GSM, the air interface is a layering of TDMA on top of FDMA to send bursts of data through a channel during a turn on a time slot. The bursts and channels vary in purpose to achieve the overall goal of being able to establish a line of communication and then provide communication services to valid MSs anywhere GSM infrastructure exists. Attackers must also provide the radio resources discussed here for MSs to find and connect to rogue base stations, using the process described in the next section.

2. Cell Selection and Reselection

The previous sections helped provide the insight into how MSs find and communicate with a network using the air interface to the BTS. However, the MS is always required to ensure it is connected to the best cell for services. This requirement is due to the likelihood that MSs will be moving at any point in time and may need resources for communication during this period of mobility. The MS periodically goes through the searching process, discussed in the Section C.1.c, and then makes a decision regarding the best cell found [18]. The searching process is also referred to as the cell selection/ reselection process. The decision-making process for cell selection and reselection first requires a collection of SIs from all the nearby cells. If the MS is already connected to a network, the current cell information is also required. The information within the SIs are then used for measurements in selecting the best cell for services [18].

MSs begin by scanning for the frequency correction bursts of known bands for service. The modem will check the SIM card for information on the last network it was connected to in order to scan for those options first. If there is no information about the network stored, then the modem will have to scan the entirety of serviceable ranges for options. Once a beacon is found, the MS will go through the normal frequency and timing alignment process to wait for the SI message from the BCCH that contains the RACH information. This process recurs for the six strongest beacons found and the MS will then sort the options based on PLMN validity and the main criterion of transmission quality. The MS will check the PLMN received from the SI against the list of FPLMNs and preferred PLMNs stored, if any, in the SIM card. The signal quality is a measurement of power of the received information signals to calculate the path loss criteria (C1) using

$$C1 = [A - \text{Max}(B, 0)],$$

where A = RLA_C – RXLEV_ACCESS_MIN

and B = MX_TXPWR_MAX_CCH – MS Max Transmitter Power.

The option with the highest C1 will likely be selected [7].

The RLA_C is a running average of at least five received signal levels at the MS. The RXLEV_ACCESS_MIN is the minimum received power threshold required for network access, sent by the BTS [18]. The MX_TXPWR_MAX_CCH is the maximum transmission power that the MS is permitted by the network to use to send information on the RACH [18]. All of these values are in decibels (dB) and the signal with the highest positive C1 will be the selected cell. The C1 criteria is primarily used for cell selection, such as when a MS is powered on or is no longer in airplane mode. However, when a MS is already connected to a cell, the C2 criteria calculated from

$$C2 = C1 + CELL\_RESELECT\_OFFSET - (TEMPORARY\_OFFSET * H(x)),$$

where x = PENALTY_TIME – T.

$$H(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

is instead used [18]. Notice that the C2 criteria take the C1 criteria into consideration; therefore, both must be measured at regular intervals for cell reselection. The new values introduced in C2 (CELL_RESELECT_OFFSET, TEMPORARY_OFFSET, and PENALTY_TIME) are provided in the SI messages broadcasted on the BCCH [7]. Once a cell is deemed to be a strong candidate, the value T is the timer value that indicates the amount of time that has passed since it was determined as such. Just as with C1, the cell with the highest positive C2 value will likely be selected [7].

All of the physical components of the GSM network have been introduced from the MS through the subsystems. The air interface that directly links the MS to the network provides the frequencies and channels for communication. However, it is up to the MS to maintain its own connection to the network as it moves from cell to cell. The specifications have primarily defined cell selection and reselection decisions to be made based off of the power levels of received BTS signals. From the literature, it remains unclear what a UE does when it cannot use power levels as a judge for cell selection and reselection procedures. The focus of our work is to examine the MSs BTS selections when the MSs receive equivalent power in order to determine if MSs have designed preferences.

D. ROGUE BASE STATIONS

A review of the GSM architecture reveals that the MS does not interface directly with any components of the GSM network beyond the BTS since all communications traverse through BTSs. This review also reveals that the MS does not authenticate a GSM network nor its components. An attacker can leverage this lack of authentication by using a rogue base station to capture IMSIs, which can then be used to track targeted devices or further malicious intentions [4]. Rogue base stations replicate authentic BTSs primarily through SDRs and open source software that mimics legitimate air traffic.

1. Software-defined radios

The creation of SDRs has turned IMSI catching from what used to be an expensive attack into a relatively cheap and easy one. SDRs are radio devices that can be configured and controlled by computer software [4]. One common SDR is the Universal Software Radio Peripheral (USRP) designed by Ettus Research [27], which can cost around $1500 [4].

The daughterboards of SDRs allow the SDRs to operate within a frequency range dictated by the daughterboard, with an example shown in Figure 10. The daughterboards can be designed to operate in one band or several of the available frequency bands. The SDRs are then configured for operation as a base station through the use of OpenBTS.

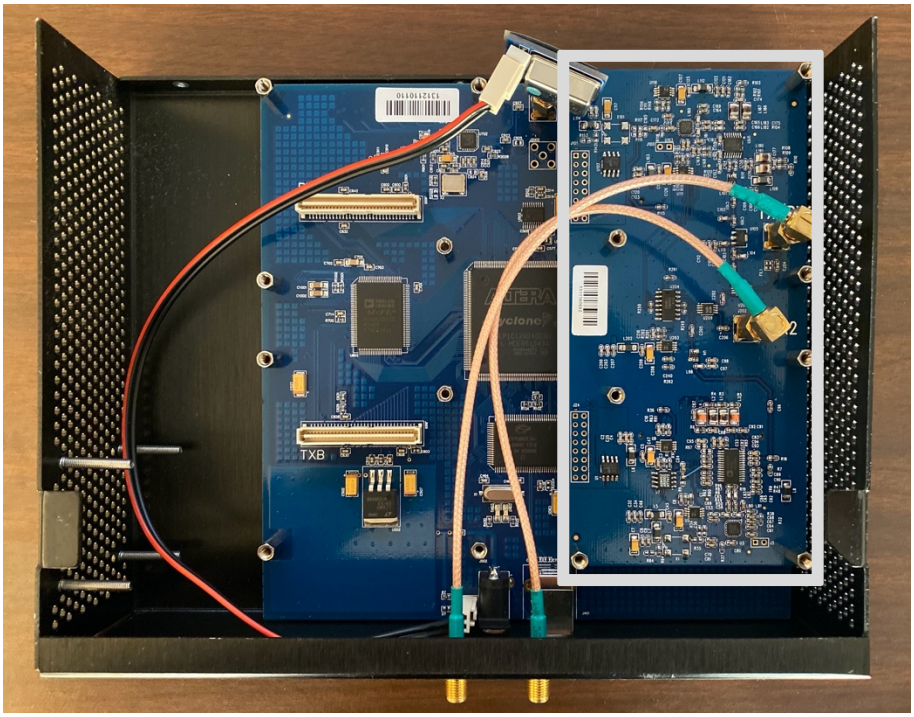

Figure 10. SDR daughterboard Outlined in White

2. OpenBTS

OpenBTS is open source software developed by Range Networks to provide a network for testing. The software was also designed to provide future opportunities for low cost cellular service options [28]. OpenBTS uses SDRs to emit all of the standard 3GPP traffic over the air interface [5]. The standard traffic includes providing combination V on C0T0 and maintaining the SI messages sending network information necessary for RACH, C1, and C2 calculations [29]. Pairing the open source software with inexpensive SDRs enables IMSI catching attacks to be carried out with cheap implementations.

3. Past Implementations

Several works have successfully created and implemented rogue base stations that illustrate many important conclusions, however, only a few will be discussed here. The similarity of the prior work proves IMSI catchers can be built using relatively inexpensive SDRs and open source software such as OpenBTS. Mruz [16] and Debrowski et. al. [17] show that devices designed for newer standards are backwards compatible to older mobile

networks. The work presented by Retterstol [4] demonstrated the ability to selectively jam a subscriber while catching an IMSI. As well, Retterstol was able to perform denial of service attacks on subscribers of a specific network while catching the IMSIs. The work provided by Weinmann [3] analyzes the ability to remotely corrupt the memory of mobile devices using rogue base stations. The authors of [2] use a similar rogue base station approach to perform attacks on mobile devices in 4G and 5G networks.

Overall, the growth of technology has both decreased the cost to implement rogue base stations and enabled more advanced threats using rogue base stations. The work in this thesis aims to leverage the inexpensive hardware implementations and uncertainty in GSM standards to determine if mobile devices have air interface preferences, enabling attackers to profile their targeted device for faster capture.

THIS PAGE INTENTIONALLY LEFT BLANK

III. EXPERIMENT SETUP

Chapter I provides the motivation for this thesis with a brief introduction to IMSI catchers, rogue base stations, and their threat to mobile subscribers. Chapter II provides the background information necessary for understanding how GSM operates through a review of each of its components and subsystems. The information in Chapter II also highlights the components of a MS used for decision making as well as the frequency and channels required for communication. This chapter presents the experiments designed to determine if the modem of a MS has frequency and channel preferences that could be used to profile a targeted device. This chapter also describes the test questions and the setup of a controlled testing environment.

A. TEST QUESTIONS

This thesis uses OpenBTS and SDRs to create IMSI catchers in a similar manner to previous work. However, we extend prior work to present a program that can configure and implement IMSI catchers on multiple base stations for specific analysis goals. The created test environment encapsulates the base stations, eliminates standard decision-making factors, and provides multiple frequency and channel configuration options using the base stations. The IMSI catching program developed for this thesis also records the configuration information of the base stations chosen by the modems to permit analysis of the decision-making patterns. If the modems have preferences, we will then work to determine whether attackers can create profiles for phone models based on three main questions:

1. Do modems have a frequency band preference?
2. Within a preferred frequency band, do modems have a channel preference?
3. Using the preferences of one and two, what are the effects on IMSI capture speeds?

The first experiment is used to determine if the MSs have frequency preferences when the available cell options are at the same power level and have no other information to help make a decision. The next experiment requires the frequency band preference revealed during the first experiment. The second experiment is designed to determine if the MSs prefer a physical channel within a frequency band.

The final experiment requires the preferences from the first two experiments to establish if the channel and frequency preferences decrease the amount of time it takes to capture an IMSI. In order for us to accomplish this, two approaches are required. The first approach uses the preferred frequency from the frequency test and the preferred channel from the channel test as the configurations, then we examine the IMSI capture speeds. The second approach randomizes channels with the four GSM bands and is again focused on examining the speeds of the IMSI catches. We will compare the speeds of the two approaches to determine if preferences make a difference in terms of speed of IMSI capturing.

B. TEST ENVIRONMENT

The testing environment is designed in a way that does not risk the privacy of legitimate users. Instead, we replicate a GSM environment within a controlled environment. The testing environment, as shown in Figure 11, is made up of two core networks to accomplish that requirement. The test network works to mimic an actual GSM network from the MSs to the BTSs. The control network provides the ability to configure portions of the network and capture the information necessary for analysis. Encapsulating both networks in a Faraday cage limits extraneous interference on the two core networks and prevents the core networks from interfering with legitimate mobile networks.

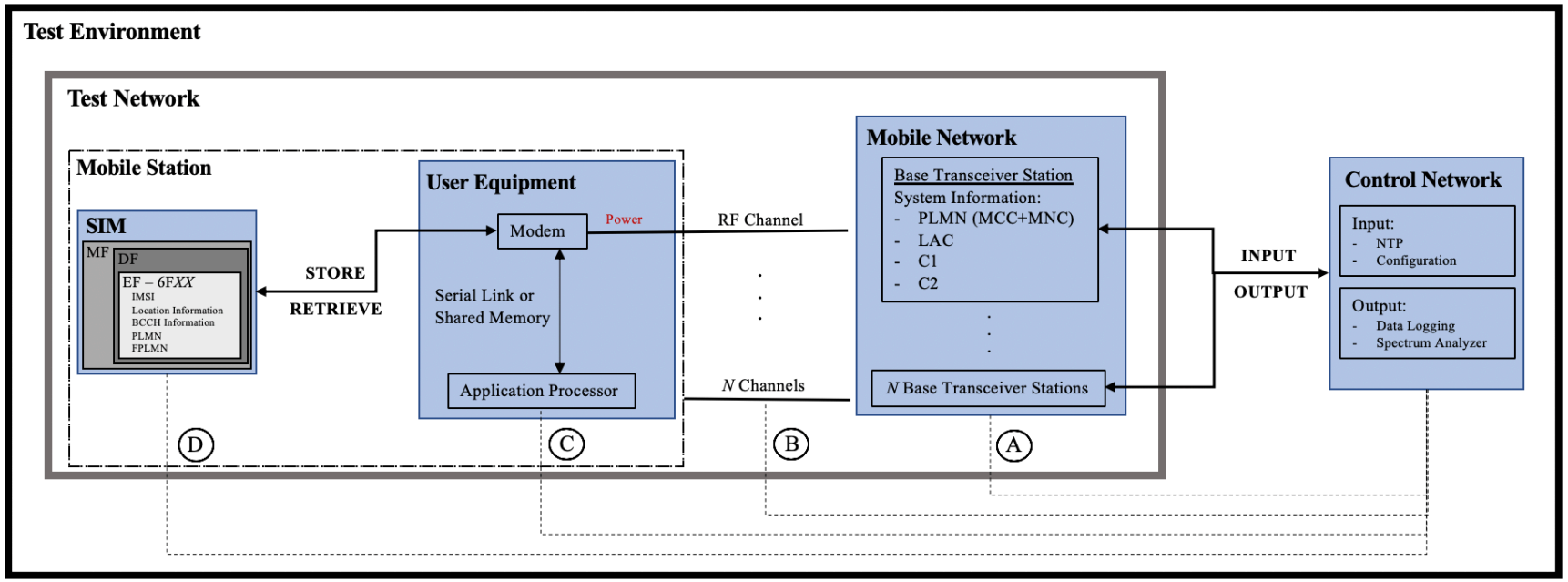

Figure 11. Experiment Environment

1. Test Network

Since the MS does not interface directly with any components of the GSM network beyond the BTS, a GSM network can be replicated by using SDRs and OpenBTS for the components shown in Section A of Figure 11. For this, we use a set of IPLinkME [30] SDRs, similar to the USRPs by Ettus Research. The daughterboards within the SDR are only able to cover the 850 MHz and 900 MHz or the 1800 MHz and 1900 MHz ranges at any given time. To reconfigure frequency band during experiments, several daughterboards of both ranges are interchanged as required. Having the multiple daughterboards of both ranges also provides coverage of all four GSM bands.

Using OpenBTS, the SDRs provide the necessary GSM network resources for MSs to connect to a network on the air interface, labeled Section B in Figure 11. The initial configuration for OpenBTS is provided in Appendix E. Some minor interfacing in Sections C and D of Figure 11 provides control for eliminating stored information that is used to make cell selection decision-making easier for the modem. Since we cannot access that information directly through the MS, a Quectel UC20 modem [31], displayed in Figure 12, is used to eliminate information in the EFs of the SIM card.

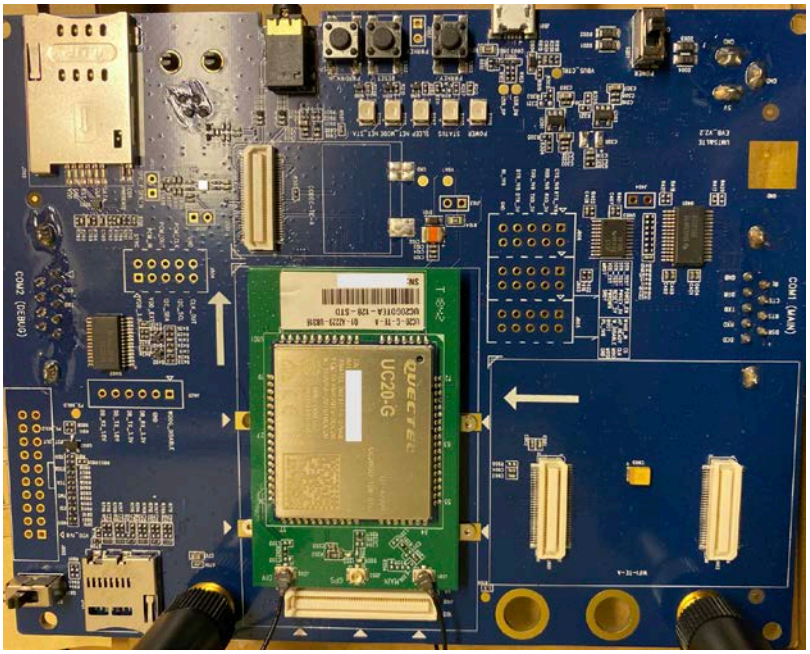

Figure 12. UC20 for SIM Configuration

For our setup the application processor is emulated using *minicom 2.7* [32] on a Linux workstation to send the AT commands that modify the necessary EFs of the SIM cards. Together, minicom and the Quectel UC20 modem represent Section C of Figure 11. The SIM cards for Section D of Figure 11 are an assortment of Range Network blank SIM cards [33]. These SIM cards are reprogrammable testing SIM cards and are configured using a combination of *pySim* [34] and *minicom 2.7*. The configurations for *pySim* and minicom are shown in Appendix A and Appendix B, respectively.

2. Control Network

The control network within the test environment has three main responsibilities: 1) configure, 2) capture, and 3) analyze. The three responsibilities are split amongst a primary workstation and the set of four secondary workstations that each interface with an SDR. The workstations are connected via a local area network (LAN) and their time is configured for synchronization using the Network Time Protocol (NTP) [35], provided in Appendix C. Using NTP ensured the workstations had as close to the same time as possible, without worrying about having the most accurate current time.

We created a set of C and python programs that configures the test network, captures the IMSI, and logs the information for analysis. The set of programs accomplish all three responsibilities for the control network. The C and Python code are provided in Appendix F for the primary workstation and Appendix G for the secondary workstations. The programs work together to supply a configuration mode and capture mode, providing the necessary data for analysis.

a. Configure

The main purpose of this mode is to configure the SDRs according to what the experiment requires. An operational flow chart for the configuration process is shown in Figure 13. Configuration mode first requires that the secondary workstations are running the script since the primary will check the status of those workstations through a health check. If the health check fails, the program will wait to provide time for troubleshooting connections. Once the health check is complete, the program, will ask if the ARFCN assignments will be randomized during configuration. Randomization for ARFCN

assignment allows for flexibility of various experimentation methods. If selected, the program only requires the user input of a valid GSM frequency band from the 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz options. Each host then receives their assigned band and the randomly selected ARFCN for that band to configure the associated SDR.

The secondary workstations each display the results of configuration in case the configuration failed. The secondary workstations then exit configuration mode and continue immediately into capture mode. However, if the ARFCN randomization option is not selected, user input for both a valid band and channel are required for each workstation to configure their SDR. The secondary workstation then goes through the same procedure of exiting and continues to capture mode.

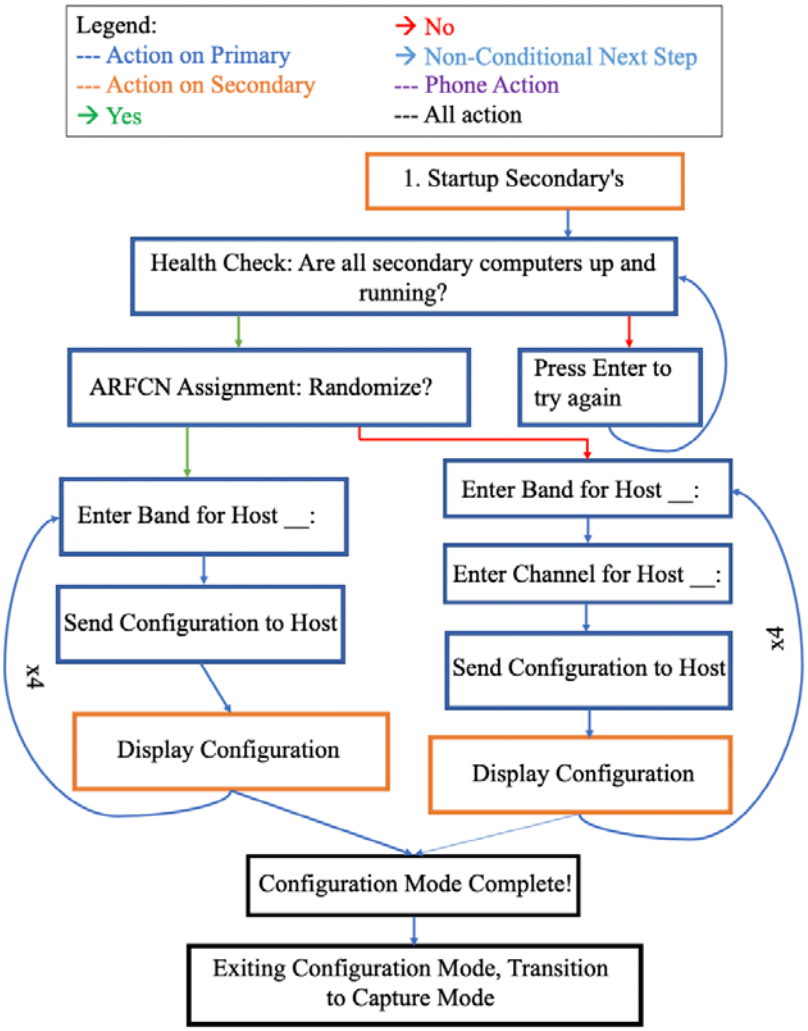

Figure 13. Configuration Mode Flow Chart

b. Capture

The primary goals of capture mode are to capture an IMSI and relay the significant data back to the primary workstation for analysis, following the operational flow chart shown in Figure 14. The program provides the ability to either directly enter capture mode without first using configuration mode or transition into capture mode from configuration mode. Having the option to go directly into capture mode is faster to use for the experiments that do not require configuration between every phone or round. Every phone will be tested individually, and a round is completed once all phones have been tested individually one time through.

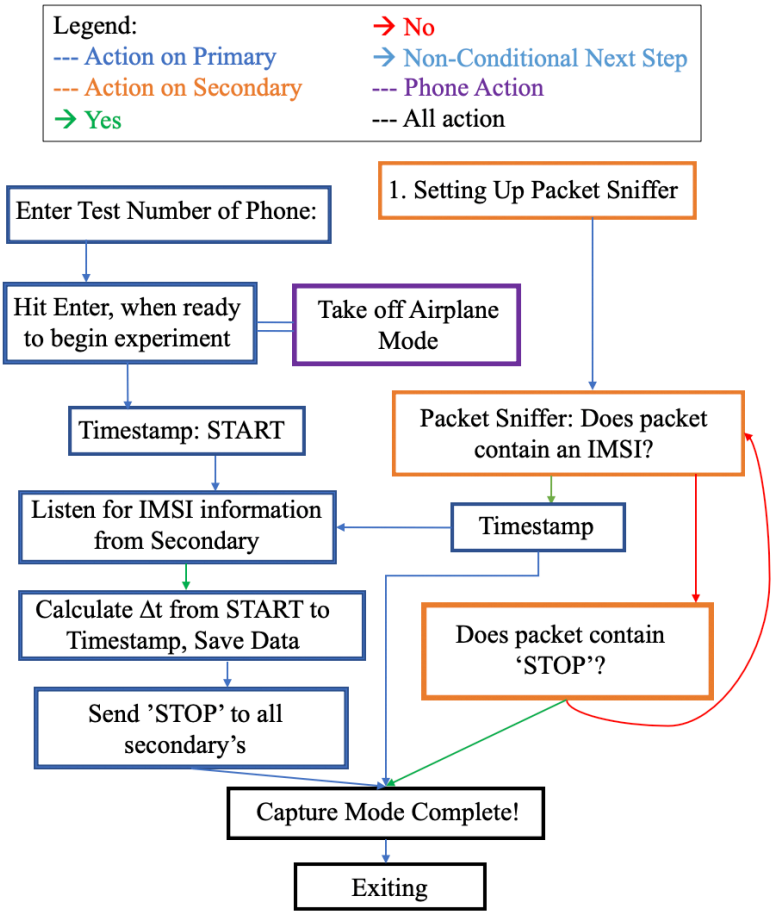

Figure 14. Capture Mode Flow Chart

Once the secondary workstations and primary workstation are in capture mode, the secondary will immediately set up the packet sniffer to sniff the air interface for the first instance of a location updating request message. The packet sniffer looks for the specific bytes to identify the correct packet on the network.

After many trials using Wireshark [36], we determined that bytes `01 3f 49 05 08` are the fingerprint for the desirable packet, with `01 3f 49` highlighted in Figure 15 and `05 08` highlighted within the same packet in Figure 16. These bytes are the minimum bytes required for ensuring the packet captured is a location updating request message of the link access procedure (LAPDm). This message contains the first instance of an MS IMSI being sent on a network. Using the minimum number of bytes also ensures that the bytes chosen will not differ by later headers in the message.

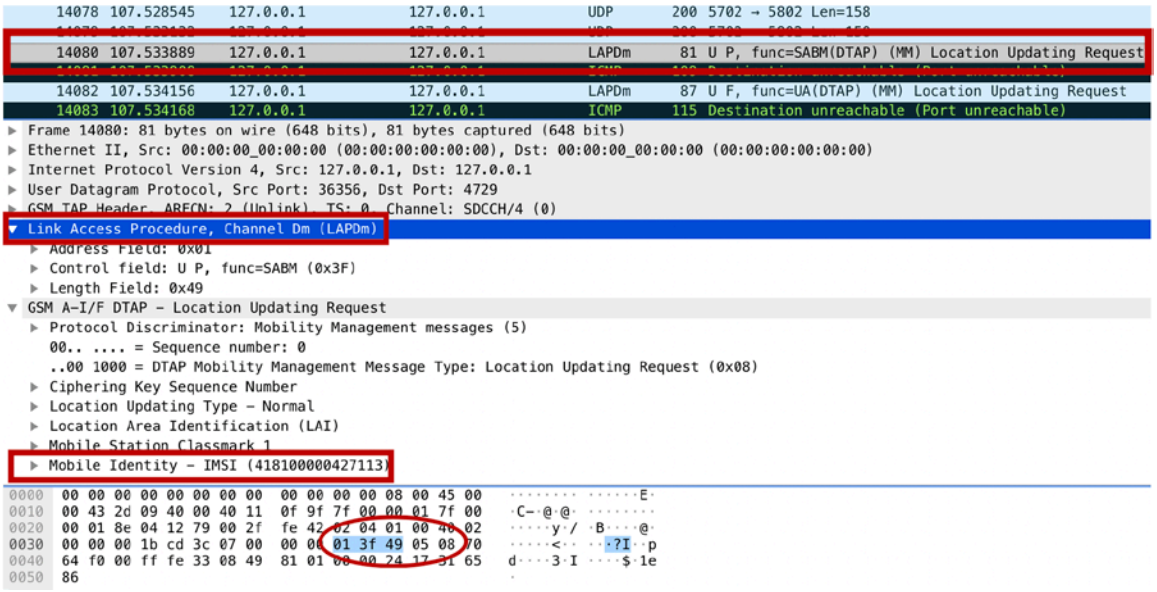

Figure 15. Wireshark Packet Capture Location Update Part I

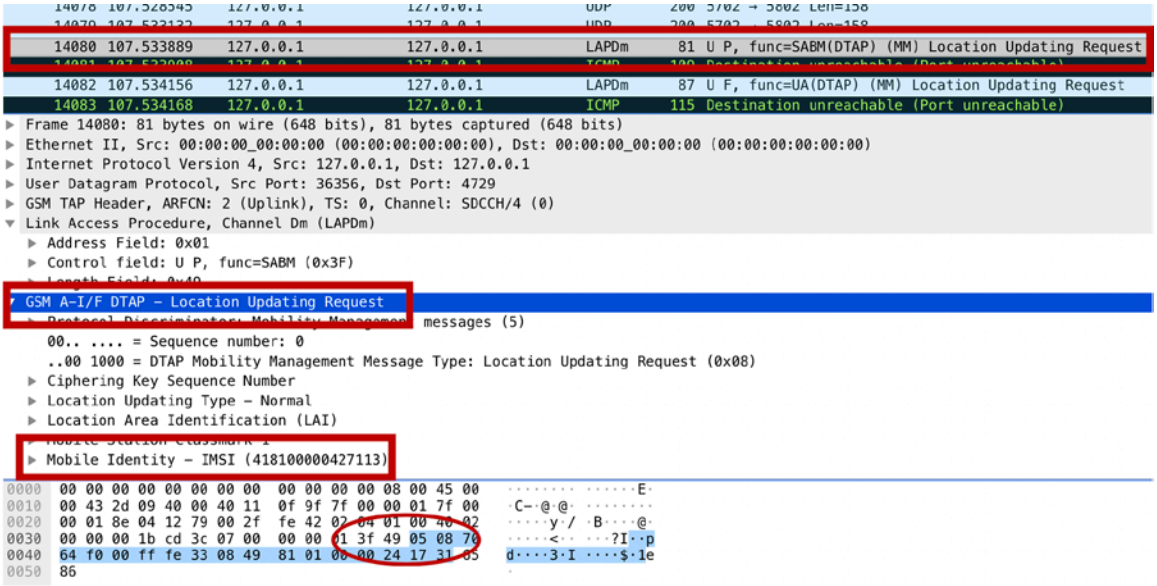

Figure 16. Wireshark Packet Capture Location Update Part II

It is important to note that while the packet sniffer is immediately set up once the secondary workstations are in capture mode, this does not mean the experimental round has begun. Prior to the experiment, the MSs for testing are placed into airplane mode to prevent any network transmissions. The devices are then labeled with a test number to

identify them in the data later on during the analysis stage. For each MS round, its associated test number is entered on the primary workstation program.

In order to initiate the round, the primary station takes in the *return* key input from the user while the user simultaneously toggles the MS off of airplane mode. These two actions start the round and allow a MS to interact with normal network transmission procedures. The program timestamps immediately when initiated to indicate the start of the round for future measurements. The primary workstation then waits for the IMSI catch data from a secondary workstation. Once one of the base stations finds the bytes indicating an IMSI has been caught, it will timestamp the capture and relay the information to the primary. This process is discussed further in Section c.

After receiving the relayed information, the primary will measure the delta time between the start time and the time stamp of the IMSI catch provided by the secondary workstation. The primary logs both the delta time and other relayed information from the secondary. For the final step, the primary will then send a "STOP" message to all of the secondary workstations to force exit their program, completing capture mode.

c. Analyze

The final requirement of the control network concerns the analysis of the information, as shown in Figure 11. The purpose of this thesis is to determine how a modem selects a specific BTS when it does not have power differences and SIM card information to help decide. A spectrum analyzer, shown in Figure 17, is used to ensure the MS sees equivalent power from the provided base station options. The spectrum analyzer is placed at the location where all base station power levels are equivalent.

Figure 17. Spectrum Analyzer

The final subset of the control network analysis section is the information that is relayed to the primary when an IMSI is caught. The secondary relays the time of the IMSI catch along with the band, channel, and IP address of the base station that captured the IMSI. The information that is relayed to the primary is then formatted and stored as a .csv file for later analysis.

To review, the test environment is contained within a Faraday cage to prohibit test network interfere with legitimate networks and prevent legitimate networks from interfering with the test environment. The test environment was further divided into the test network and the control network. The test network recreates a GSM network while the control network accesses portions of the test network for experimentation and analysis.

C. EQUIPMENT

This final section of Chapter III provides a listing of the equipment used throughout the experiments. In summary, the test environment equipment used against the test phones within the Faraday cage included:

- 1 Primary Workstation: HP ProBook PC, Linux Ubuntu 16.04 LTS
- 4 Secondary Workstations: Asus Notebook PC, Linux Ubuntu 12.4 LTS
- 4 IPLinkMe SDRs

- 1 Switch and 5 Ethernet Cables for the LAN
- 1 Spectrum Analyzer: Agilent Technologies MXA Signal Analyzer, N0902A
- 1 Quectel UC20 Dedicated to Template SIMs

with various phone models discussed in Chapter IV. Figure 18 displays the equipment used in a workspace diagram.

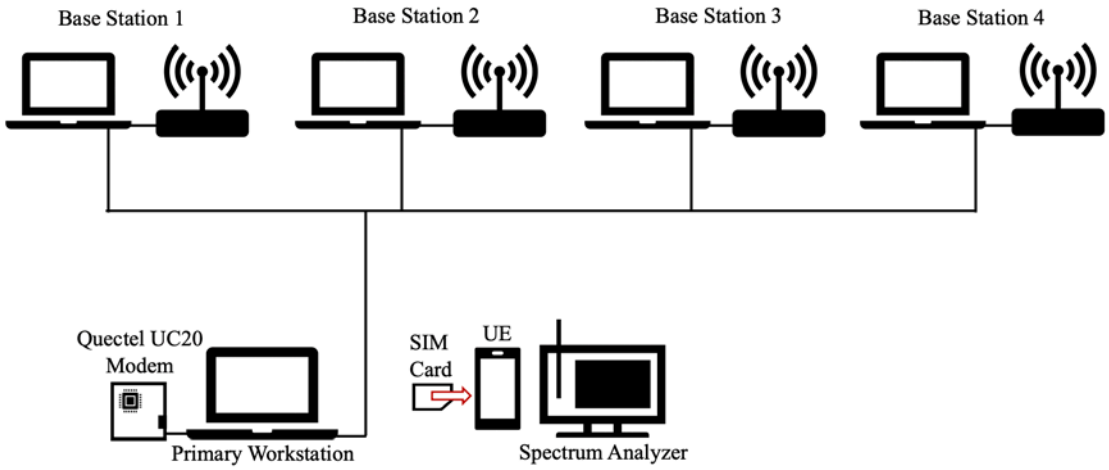

Figure 18. Workspace Diagram

IV. RESULTS

Chapters I and II provide the purpose and background for this thesis. Chapter III explains the questions that influenced the design of the test environment and details the programs and equipment used by the test environment. This chapter explains the pre-experiment dry runs used to solidify the test environment in addition to the experiments designed in Chapter III. For the first experiment, we cover the objective, process, and results of the frequency band test. Next, we detail the objective, process, and results of experiment two, the channel test. Finally, we use the frequency test and channel test preferences from experiments one and two to configure and execute our third experiment. In experiment three, we attempt to compare the speed of IMSI catching using the preferences against the speed of IMSI catching using random valid assignments.

A. PRE-EXPERIMENT TESTING

The purpose of the experiments is to determine if MSs have a frequency and channel preference for cell selection. To accomplish our experiments, we first have to establish a range of phone manufacturers for testing. Then we must work to eliminate and control the EF binaries that affect cell selection decision-making throughout the experiments. Additionally, the MSs must sense equivalent power from all base stations to minimize decision-making based on receiver power and reveal any frequency preferences. Finally, we work to minimize the time per round for each experiment so we can maximize the number of rounds that can be performed as well as the number of test devices that will be used for experimentation.

1. UE

The experiments in this thesis first require procuring a set of devices that represent a range of manufacturers as well as different options within the same manufacturer family. The list of devices acquired for our experiments is shown in Table 3. This table includes devices of different manufacturers as well as devices that are from the same manufacturer but different models. Additionally, we include devices that are from the same manufacturer and model family to see if manufacturers may change their modem design based on the

region of service. Region-based modem differences may also include hardware or software upgrades.

Table 3. Pre-experiment UE Devices

Phone Manufacturer	Model
Samsung	Galaxy SII—English
Samsung	Galaxy SII—Spanish
Huawei	Honor
Apple	iPhone 3
Apple	iPhone 6
Quectel	UC20

The Samsung Galaxy SII phone is an example of a manufacturer that created different options within the same model family. Samsung appears to have made at least two different options within the Galaxy SII model, according to the labels displayed in Figure 19. We discovered the labels behind the batteries inside of the phones were printed in two different languages, Spanish and English. Use of the phones reveals no obvious differences and the manufacturing labels indicate they are the same GT-I9100 model. We chose this Samsung Galaxy SII (SGSII) model with the English and Spanish labels to see if either manufacturing location or intended use location cause differences in cell selection.

Figure 19. Samsung Galaxy SII: Spanish vs. English

The Apple iPhone 3 and iPhone 6 model types allow us to test if band and/or channel preferences can change when a manufacturer upgrades the hardware and software of their devices. The Huawei Honors extend our range of cellphone manufacturers tested while the Quectel UC20s expand the scope of this research to include testing a cellular IoT device.

2. SIM Card Binaries

Several trial rounds revealed that the EF binaries changed after cell selection. An example shown in Figure 20 displays many of the EFs discussed in Chapter II. The examples shown are produced using minicom and seeSIM.txt from Appendix B.B.

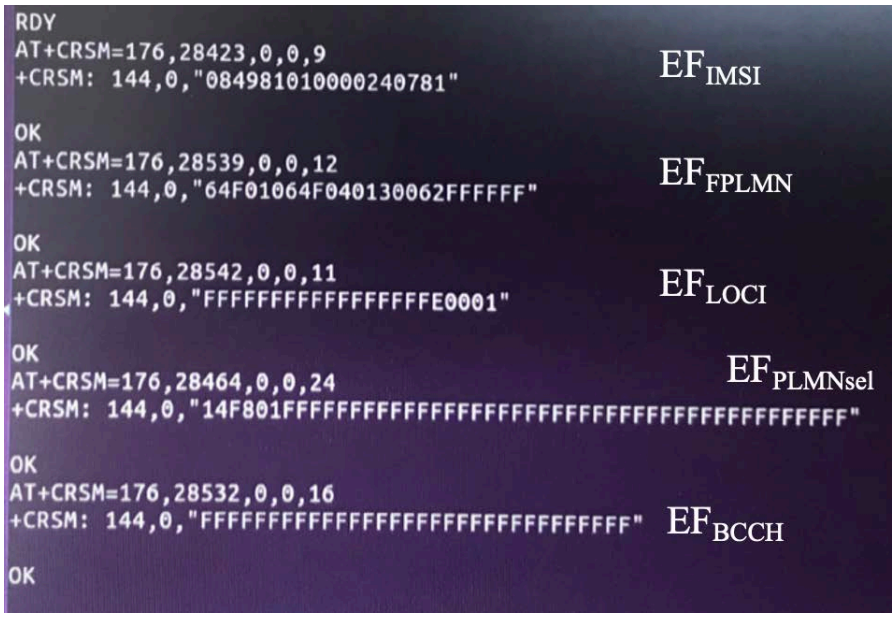

Figure 20. seeSIM.txt EFs

Five test rounds using three of each of the devices in Table 3 were used to determine the EFs that are updated after cell selection, with the results in Table 4 and an example of EF changes in Figure 21. The results from the five test rounds show that all three Huawei Honors were affected every round. Two out of three Quectel UC20 devices were updated every round while one SGSII Spanish device was affected by only one round. The Apple and SGSII English devices were not affected by any of the five test rounds.

Table 4. SIM Cards Affected Five Rounds

Number of Models Affected	Model Affected	Number of Rounds Affected	Areas Modified
1 of 3	Samsung Galaxy SII—Spanish	1 of 5	LOCI
3 of 3	Huawei Honor	5 of 5	BCCH and LOCI
2 of 3	Quectel UC20	5 of 5	LOCI

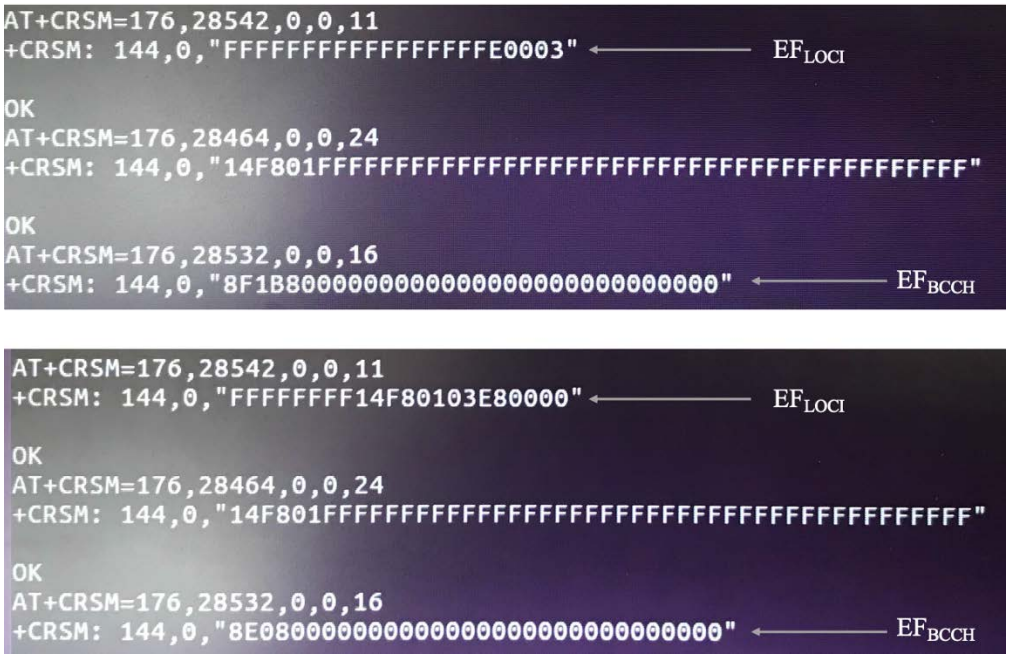

Figure 21. Two Examples of EF_{BCCH} and EF_{LOCI} Post Cell Selection

The results of Table 4 confirm that the information within the SIM card is affected by cell selection. To ensure consistency, we chose to refresh the SIM cards after every round even though every device did not appear to update every round nor was every EF updated during cell selection each time. The tempSIM.txt script to template (i.e., reformat) each SIM every round is provided in Appendix B.A, with an example of the template shown in Figure 22.

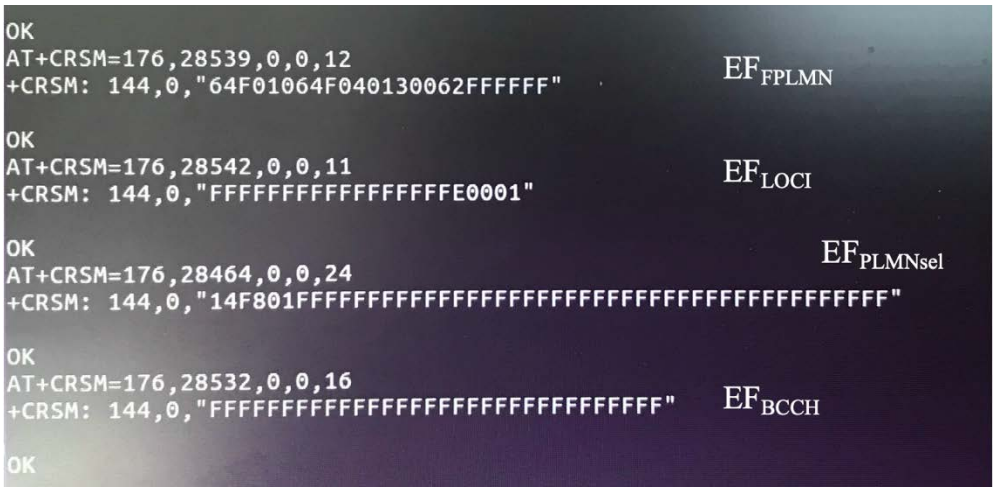

Figure 22. Template for SIM Cards

The binary for EF_{FPLMN} had been found on one SIM card and since the FPLMN information listed is not equivalent to the test PLMN in this experiment, this binary was used for all the SIM cards in the template. The bytes of EF_{LOCI} and EF_{BCCH} were nulled to ensure the locations could not assist in cell selection decision-making processes. Lastly, the $EF_{PLMNsel}$ only contained the PLMN of the test network, to ensure the test network would not be blocked during experimentation.

3. Power Measurement, Harmonic Suppression, and Attenuation

Since power is a factor considered for cell selection, equivalent power must be shown at the spectrum analyzer where the MS remains throughout each round. Upon inspection of the spectrum analyzer with the Faraday cage door shut, it was apparent that the power levels required an adjustment at the SDRs. The SDR frequency assignments resulting in Figure 23 are 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz, one for each SDR. With only four SDRs in use, only four peaks are expected instead of the six shown in Figure 23.

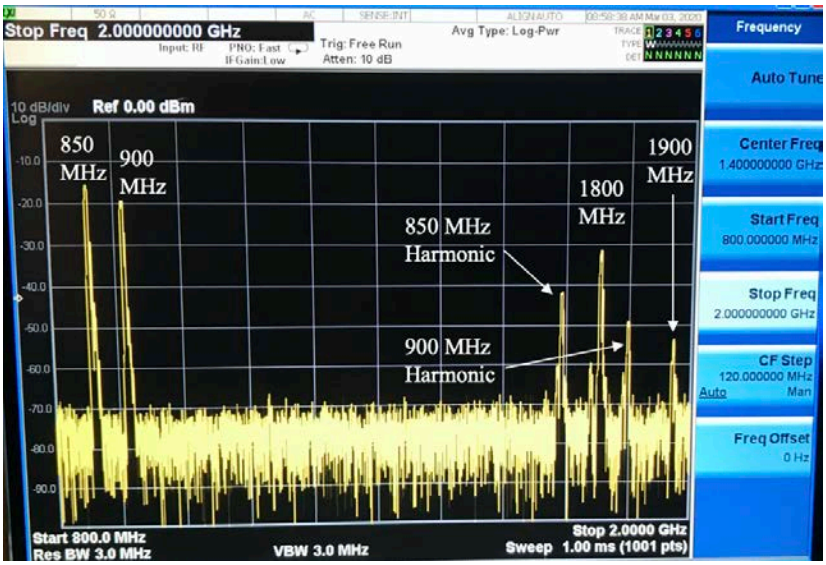

Figure 23. Unfiltered and Unattenuated Downlink Spectrum

The 850 MHz and 900 MHz SDRs produced harmonics causing the extraneous peaks seen in Figure 23. Using a low-pass filter (LPF) on both the 850 MHz and 900 MHz SDRs eliminated the harmonics and results in the spectrum shown in Figure 24.

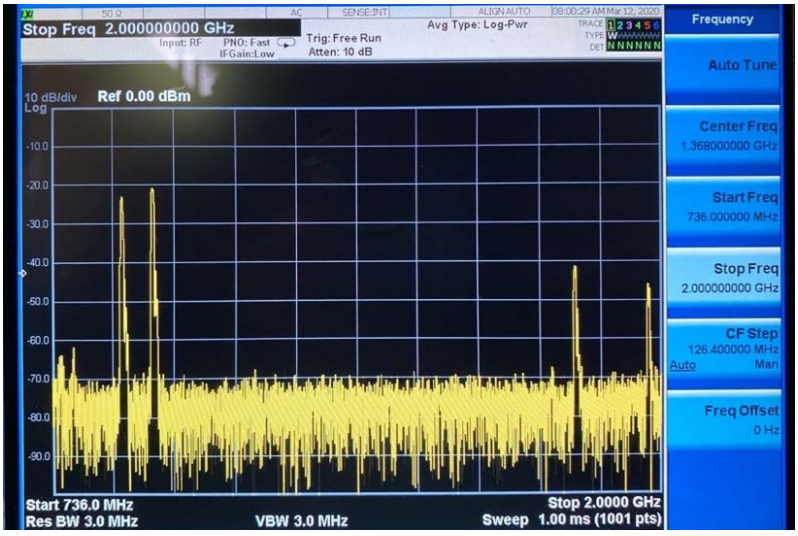

Figure 24. Filtered, Unattenuated Downlink Spectrum

The resulting power peaks in Figure 24 show the 850 MHz and 900 MHz SDRs are approximately 20 dB higher than the 1800 MHz and 1900 MHz SDR peaks. A set of 6 dB

and 10 dB attenuators reduced 850 MHz and 900 MHz peaks to approximately the same height of the 1800 MHz and 1900 MHz peaks. The resulting spectrum is shown in Figure 25 while the application of the LPF and attenuators on the receiving antenna of an SDR is shown in Figure 26.

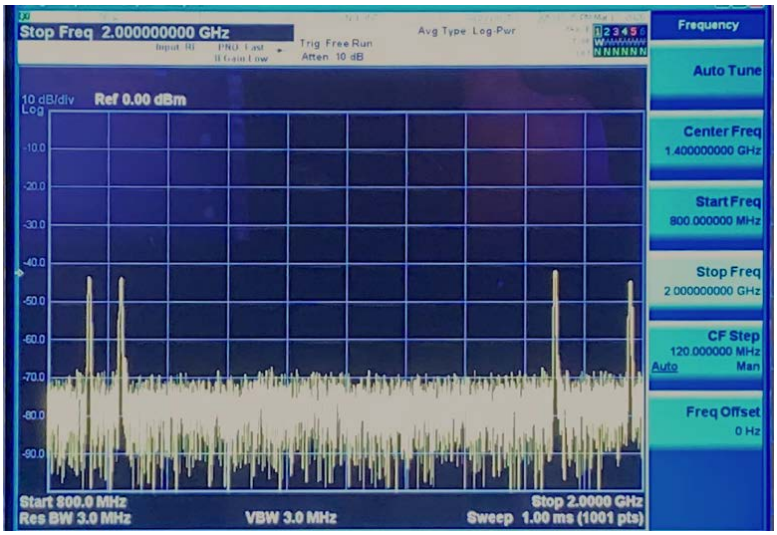

Figure 25. Filtered and Attenuated Spectrum

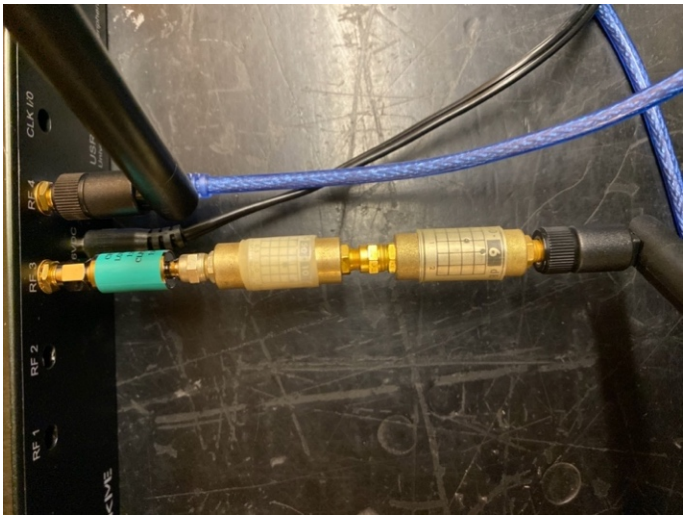

Figure 26. LPF (Green) and Attenuators

While Figure 25 depicts closely matched peaks, the power peaks continued to fluctuate throughout the experiments, resulting in peaks that differed by as much as 10 dB, as shown in Figure 27.

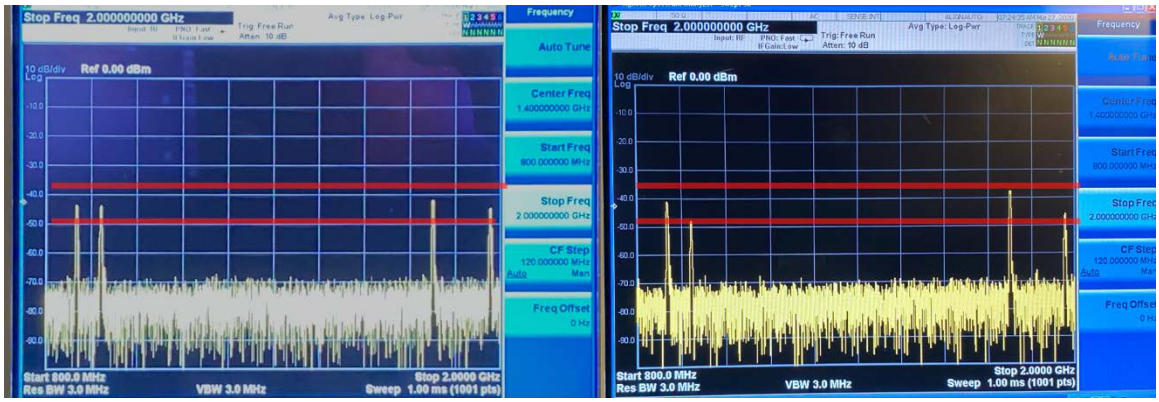

Figure 27. Matched vs. Unmatched Downlink Power Peaks Observed During Experiments

This result fell slightly short of our objective to get the power levels matched for the MS. However, due to COVID restrictions and equipment purchase timelines, we were unable to further level the power at the SDRs.

4. Time Constraints

The final requirement before experimentation was to determine the number of devices that can be used for testing and the number of rounds that can be achieved per experiment. We chose to limit our examination to three devices of each model due to limited test equipment of certain models. Each device was assigned a test number, provided in Table 5.

Table 5. Phone Models and Test Numbers for Experiments

Phone Manufacturer	Model	Test Numbers
Samsung	Galaxy SII—English	1–3
Samsung	Galaxy SII—Spanish	4–6
Huawei	Honor	7–9
Apple	iPhone 3	10–12
Apple	iPhone 6	13–15
Quectel	UC20	16–18

We then tested each phone to ensure that every device could connect to each of the four bands. Every round with the 18 MSs would take approximately 45 minutes of constant hands-on time to accomplish. We were required to spend approximately 15 of the 45 minutes to refresh all of the SIM cards after every round using the procedure detailed in Figure 28. Therefore, we were limited to ten rounds per experiment in order to accomplish an experiment over the course of a day. Before beginning each experiment, we also allotted for an additional ten minutes to ensure the clocks of the workstations properly synchronized via NTP.

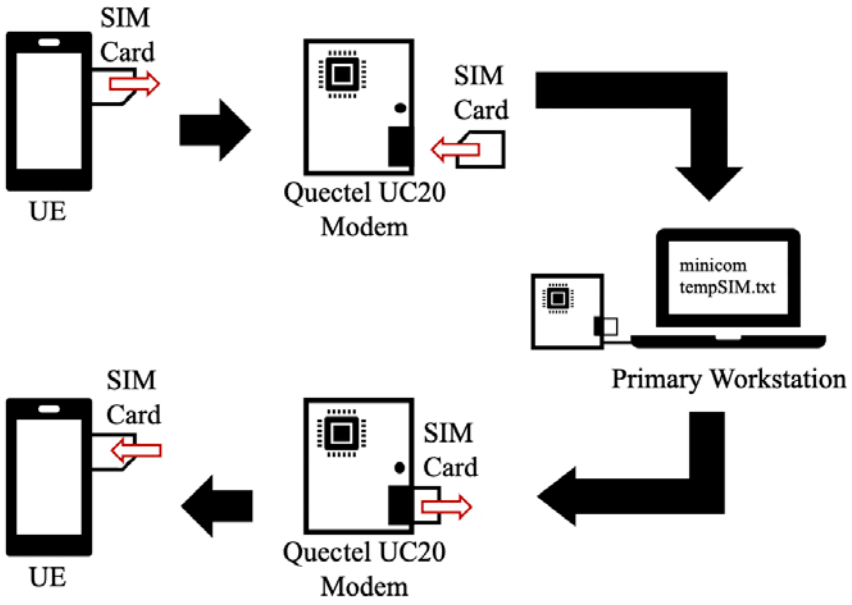

Figure 28. SIM Card Template Procedure

B. EXPERIMENT 1: FREQUENCY BAND TEST

The objective of the frequency band test was to determine if a cellular device has a frequency band preference when all of the valid GSM frequency band options are available. The frequency band test provides the spectrum of GSM frequency bands as base station options for the MSs with the experiment configurations displayed in Table 6.

Table 6. Experiment 1 Configurations

Experiment 1: Frequency Band Test	
Band Configuration	850 MHz, 900 MHz, 1800 MHz, 1900 MHz
Channel Configuration	Randomize Channel Assignment, Beginning
Round 1	Configuration Mode (Band, Channel), Capture Mode, Template SIM
Rounds 2–10	Capture Mode, Template SIM

The selection preferences during the frequency band test, provided in Figure 29, are clear for only certain test devices. The bar graph results indicate the SGSII English devices selected 1800 MHz at least 90% of the time in comparison to the SGSII Spanish devices where only one third of the devices chose the 1800 MHz band. Two thirds of the SGSII Spanish devices instead favored the 900 MHz band.

The iPhone 3 devices did not appear to favor a certain band. The iPhone 6 device, phone 14, chose the 900 MHz band all ten rounds of the frequency band test. Using [37] and device serial numbers, we discovered phone 14 was manufactured at a different factory in China than the other two iPhone 6s. We cannot definitively conclude the selections by phone 14 are a direct result of being manufactured in a different factory than phone 13 and 15. However, the results suggest that manufacturing location could be a contributing factor for cell selection decision-making. It is important to note that phone 15 for the iPhone 6 only has nine data points instead of ten. Phone 15 had been mistakenly used in only nine rounds and the mistake was not noticed until post-experiment analysis.

The Huawei Honor devices favored the 850 MHz band approximately 60% of the time for two out of the three devices while the third device only selected the 900 MHz band. The UC20 devices have the clearest selections since two out of the three only chose the 1900 MHz band while the third device only chose the 1800 MHz band.

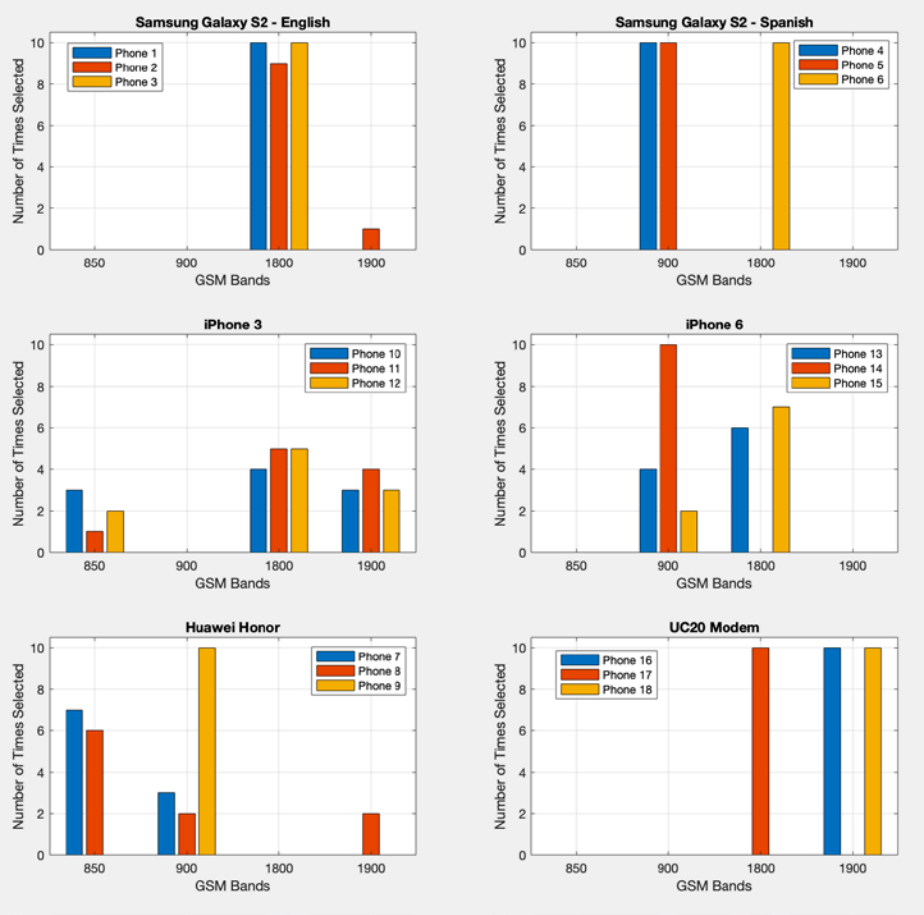

Figure 29. Experiment 1 Selection Results

The capture times for Experiment 1, shown in Figure 30, suggest possible relationships to the band selections. The SGSII English and iPhone 6 devices all show a large capture time during the first round of the frequency band test. Phone 14 is again inconsistent in that its capture time for round one is lower than the other two iPhone 6s capture times for round one. However, the capture times for the SGSII English and iPhone 6 devices significantly decrease by the second round and plateau for the remaining rounds.

Only phone six of the SGSII Spanish devices followed that pattern, additionally it is the only SGSII Spanish device that selected the 1800 MHz band. The SGSII English also favored the 1800 MHz band which suggests that the band selection may be related to capture times. However, every device that also selected the 1800 MHz band did not always experience the high first round time which could imply this relationship differs between

manufacturers. Phone ten is the only iPhone 3 device that experienced high capture times for seven of the ten rounds. Similarly, phone nine is the only Huawei Honor device that contributed to high capture times for all ten rounds.

It's important to clarify the higher average times observed for the UC20s in Figure 30 are likely due to the power-on time for the devices. The UC20 modems do not have an airplane mode that can be used to prevent cellular network transmissions while keeping the device powered on. For the UC20 modems, the start time for a round would begin when the device was powered and was the only device to differ in the round procedure.

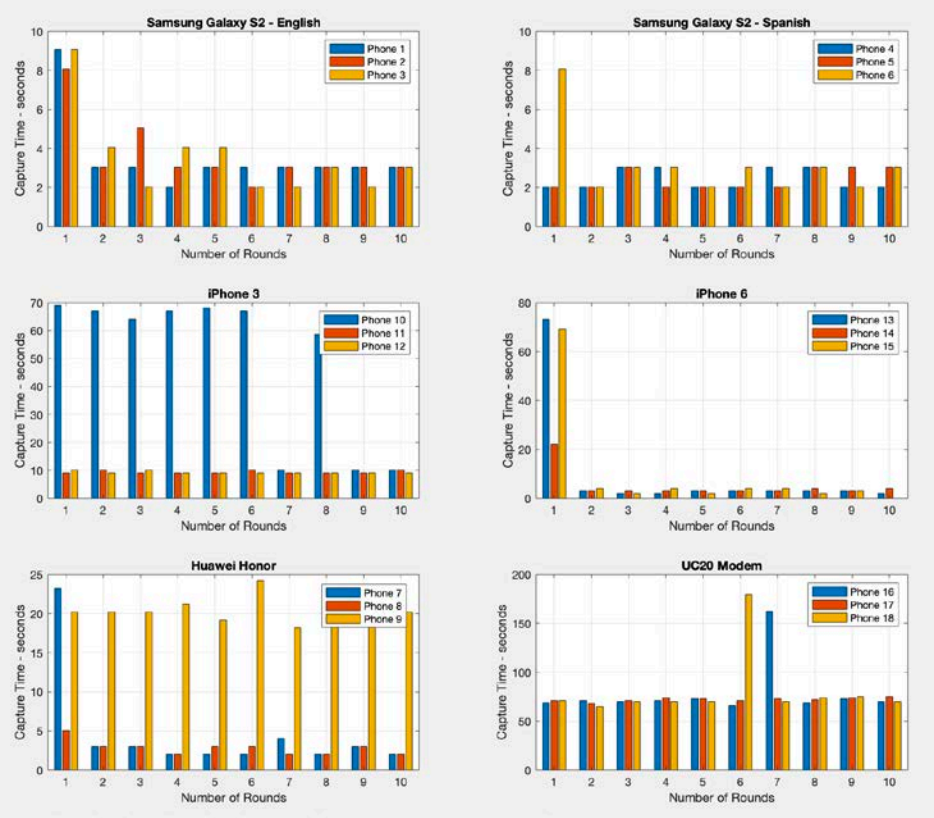

Figure 30. Experiment 1 Capture Time Results

The information in Table 7 summarizes the total number of selections for each band during Experiment 1. The data does not clearly reveal whether the phones prefer the 1800 MHz band or workstation two. The lack of differentiation suggests a mistake in the

experiment design that could reveal clearer conclusions if the four base stations would have been randomly assigned to a frequency band.

Table 7. Experiment 1 Band Occurrences

Workstation	Band	Number of Occurrences
4	850 MHz	19
3	900 MHz	51
2	1800 MHz	76
1	1900 MHz	33

Some devices during the frequency band test showed signs of having a frequency preference when presented with the GSM frequency options. Performing more rounds and a new setup that randomizes which base station gets assigned a band could help to clarify whether the modems of cellular devices have a frequency preference. Due to time constraints we were unable to perform more rounds or revise the experiment design. Instead, we chose to randomize base station configurations for the subsequent experiments.

C. EXPERIMENT 2: CHANNEL TEST

Based on the specifications, it is not clear how a MS chooses a channel when a base station presents the MSs with a set of channel options. Some possibilities on how a MS chooses a channel include: selecting channels from low-to-high (or high-to-low) within the ARFCN range or randomly or pseudo-randomly selecting a channel from the options presented.

The primary objective of the channel test was to reveal if a cellular device has a channel preference within a frequency band. Since the frequency band test did not reveal conclusive preferences, we chose the 900 MHz band as the frequency for the channel test. The power spectrum for the remaining experiments using the 900 MHz band shown in Figure 31 also identified power fluctuations of approximately 10 dB between the smallest and largest SDR peaks.

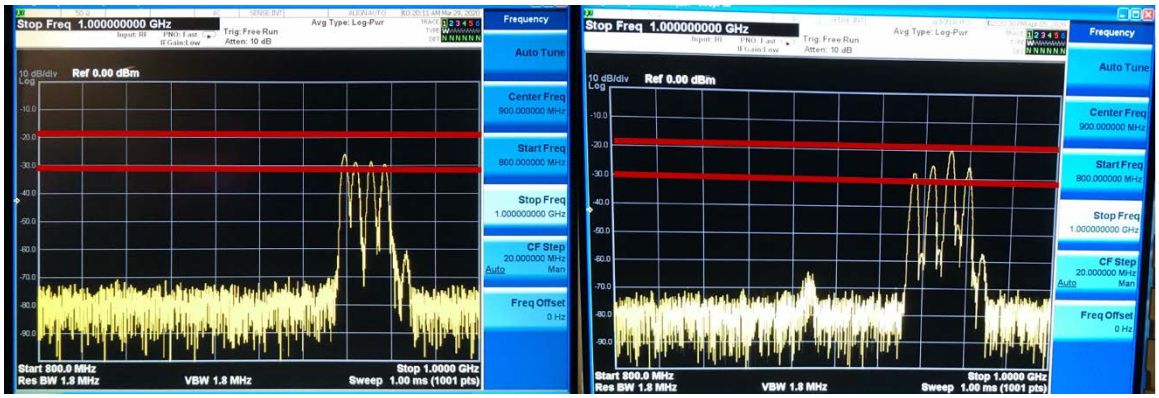

Figure 31. Matched vs. Unmatched Downlink Power Peaks, 900 MHz Band

The channel test divides the available 900 MHz channels into four evenly spread channel assignments, with the configurations displayed in Table 8. The purpose of evenly spreading the channel assignments is to test if modems favor a certain portion of the frequency band when provided a range of options within a specific band. For example, if a device favors the higher end of an ARFCN range, this may suggest the modem selects channels from the highest to lowest ARFCN.

Using *randChannel.py* in Appendix F.C produced the base station channel assignments in Table 9. During post experiment analysis we determined the channel assignment distribution, shown in Table 9.

Table 8. Experiment 2 Configurations

Experiment 2: Channel Test	
Band Configuration	900 MHz
Channel Configuration	1, 42, 83, 124
Round 1	Configuration Mode (Band, Channel), Capture Mode, Template SIM
Rounds 2–10	Configuration Mode (Channel), Capture Mode, Template SIM

Table 9. Experiment 2 Channel Assignments

Round	Workstation 1	Workstation 2	Workstation 3	Workstation 4
1	42	124	1	83
2	83	1	42	124
3	83	124	1	42
4	124	83	42	1
5	1	124	83	42
6	42	83	1	124
7	124	1	42	83
8	124	42	83	1
9	1	42	124	83
10	42	124	83	1
Channel	**Number of Occurrences per Workstation**			
1	2	2	3	3
42	3	2	3	2
83	2	2	3	3
124	3	4	1	2

The results of the channel test are clearer for certain devices as shown in Figure 32. Two out of the three SGSII English devices chose ARFCN 124 approximately 80% of the time while only one third of the SGSII Spanish devices selected ARFCN 124. The SGSII English devices favored the 900 MHz band in Experiment 1 which suggests that devices may have channel preferences across the frequency bands. The remaining two SGSII Spanish devices were inconsistent with one favoring ARFCN 83 and the final device only selecting ARFCN 42.

The iPhone 3 devices selected the ARFCN 124 approximately 60% of the time and ARFCN 83 the rest of the time. The results for the iPhone 6 show that two of the three devices selected ARFCN 124 100% of the time; this differs from the iPhone 3 results.

Phone nine for the Huawei Honor is the only device that showed a preference, selecting ARFCN 1 every round. Furthermore, phone nine is the only Huawei Honor device that favored a band and chose the 900 MHz band all ten rounds. The UC20 modems are the devices with most distinct selections for Experiment 2. For all ten rounds, two of the three UC20 modems only selected ARFCN 124 and the remaining device only selected ARFCN 42.

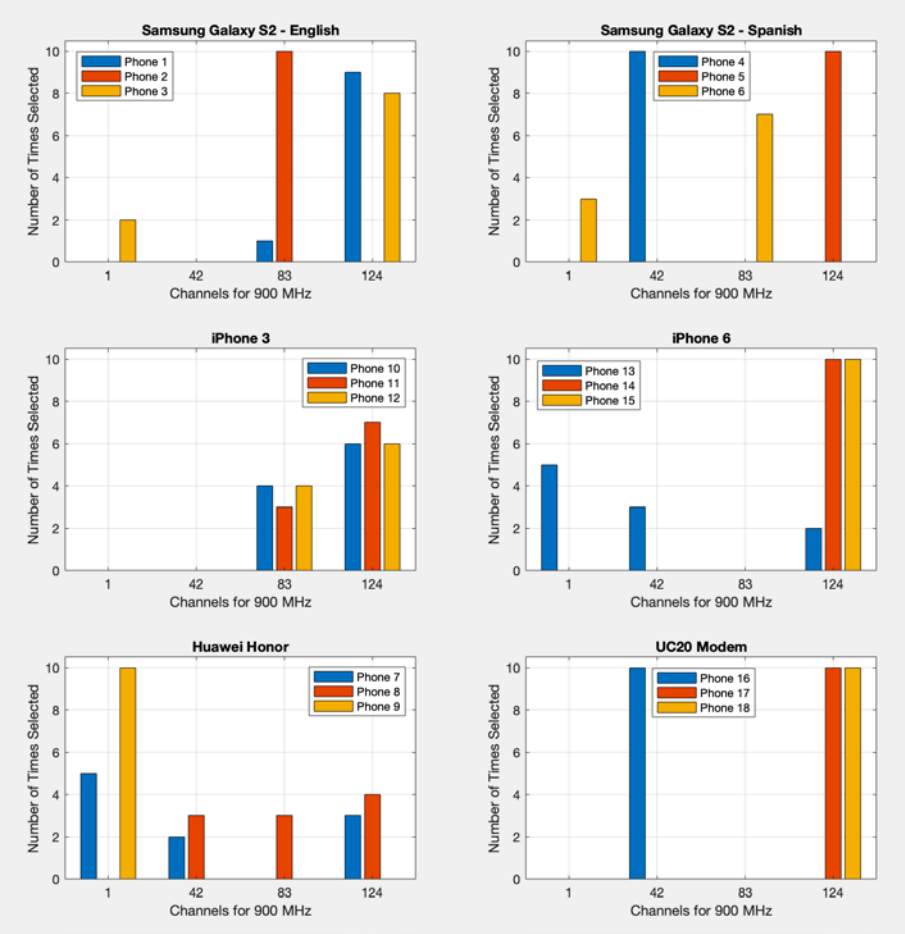

Figure 32. Experiment 2 Selection Results

The capture times displayed in Figure 33 show that the devices in Experiment 1 with high capture times during the first round did not have high capture times for the first round of Experiment 2. The devices that had high capture times during various rounds of Experiment 1 also did not have high capture times during those rounds in Experiment 2.

The round times could be related to the frequency bands chosen for certain manufacturers. The SGSII English and phone six for SGSII Spanish primarily selected the 1800 MHz band. Those devices had higher first round times than the SGSII Spanish phones four and five which selected the 900 MHz band and did not have higher first round times. Figure 33 shows that all of the SGSII Spanish and English devices have similarly low capture times for all ten rounds using the 900 MHz band.

Many of the Apple devices that chose 900 MHz their first rounds had higher capture times but did not show the same results in Experiment 2. The two devices for the iPhone 3 that did choose the 1800 MHz band their first round, did not have higher capture times for any of the ten rounds during Experiments 1 and 2. This suggests there is a relationship between the bands chosen and the capture times between models of the same manufacturer. Furthermore, the results also suggest that the manufacturer for the phones can make a difference in band selection. The higher capture times the first round but not subsequent rounds may indicate cell information is additionally stored elsewhere on the MS. This could mean the information is stored in EFs that we did not account for or that the information is stored on the UE in non-temporary storage, since the devices were powered down after every round.

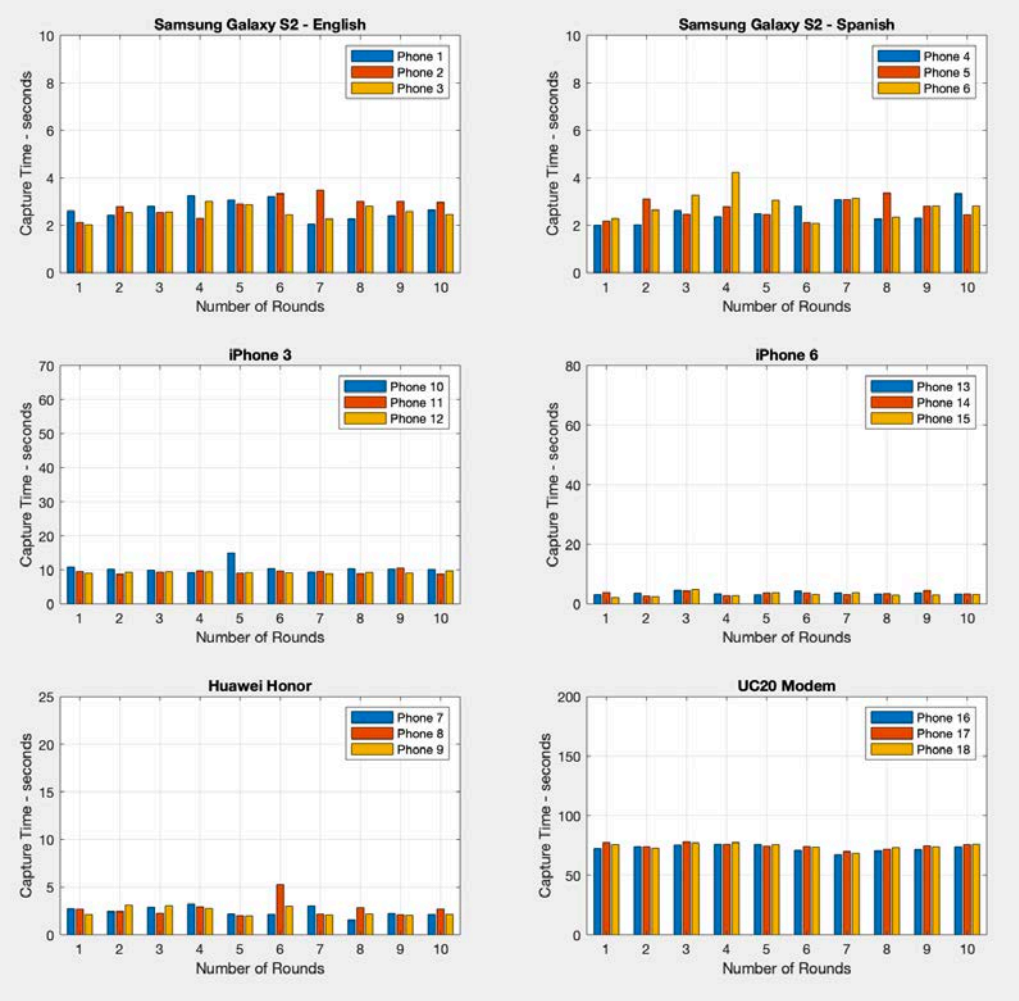

Figure 33. Experiment 2 Capture Time Results

The results from Experiment 2 are similar to Experiment 1 in that some devices show signs that suggest they have preferences. For this experiment, we randomly assigned the four channel options to the base stations to ensure the MSs do not make selections favoring a base station. The channel selection occurrences provided in Table 10 show ARFCN 124 was favored over the other channel options. According to the workstation selection occurrences in Table 11, workstation two was selected the most. When we consider the channel assignments of Table 9, workstation two was assigned ARFCN 124 four times for the experiment. This result suggests that ARFCN 124 could have been favored since workstation two was assigned ARFCN 124 more than the other workstations.

Table 10. Experiment 2 Channel Selection Occurrences

Channel	Number of Occurrences
1	25
42	28
83	32
124	95

Table 11. Experiment 2 Workstation Selection Occurrences

Workstation	Number of Occurrences
1	42
2	55
3	44
4	39

Due to the lack of clear results, we decided to explore the channel test further via Experiment 2.1 to determine if MSs have a preferred channel or a high-to-low/low-to-high range preference in the remaining experiments.

D. EXPERIMENT 2.1: CHANNEL TEST

The intention of this experiment was to test the channel randomization method to establish if a MS will select the highest channel that is offered by a base station. We simplified the configuration setup by focusing on one device in an attempt to maximize the ten rounds during the experimentation time available due to COVID restrictions.

Due to their relative ease of configuration, for the remaining experiments we used ten Quectel UC20 modems in place of the other devices used in the prior experiments. The base stations were then configured according to Table 12 with the channel assignments per Table 13. The channel assignments in Table 13 were randomly assigned during the

configuration process using the randomization component contained in *primary.c* in Appendix F.A.

Table 12. Experiment 2.1 Configurations

Experiment 2.1: Channel Test	
Band Configuration	900 MHz
Channel Configuration	Randomize Channel Assignment, Every Round
Round 1	Configuration Mode (Band, Channel), Capture Mode, Template SIM
Rounds 2–10	Configuration Mode (Channel), Capture Mode, Template SIM

Table 13. Experiment 2.1 Channel Assignments

Round	**Workstation 1**	**Workstation 2**	**Workstation 3**	**Workstation 4**
1	2	45	73	97
2	86	4	56	111
3	27	44	65	96
4	19	49	65	98
5	66	21	39	123
6	24	37	90	103
7	71	33	17	109
8	55	30	77	109
9	25	46	85	115
10	45	16	68	102

The results for Experiment 2.1 suggest that the devices favored the workstation three base station. The breakdown of workstation selections is shown in Table 14. The

results show that the modems did not favor the highest or lowest channel of the options provided. The modems instead selected workstation three 76% of the time. We did not identify other preferences during Experiment 2.1.

Table 14. Experiment 2.1 Number of Workstation Occurrences

Workstation	Number of Occurrences										
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	TOTAL
1	0	1	0	1	1	0	0	0	1	1	5
2	4	0	0	2	1	2	3	3	3	0	18
3	6	9	9	7	8	8	7	7	6	9	76
4	0	0	1	0	0	0	0	0	0	0	1

The results from this experiment did not reveal how MSs select channels during cell selection, due to non-uniformly distributed channel assignments. The randomization component of *primary.c* did not have uniform configuration assignments and this component does not check for uniform distribution. The randomization option for configuration mode provides the configurations before the start of every round and not prior to beginning the experiment. The primary workstation coincidentally did not assign a lower number for workstation four, as shown in the assignments within Table 13. As well, the primary workstation assigned workstation three the mid-high number almost every round. For the final experiment, we redesigned the setup to ensure the channel assignments were uniformly distributed.

E. EXPERIMENT 2.2: CHANNEL TEST

Experiment 2.2 was designed to expand the configuration options from Experiment 2, using some of the lessons learned during Experiment 2.1. The purpose of this experiment was to explore the channel test further using the same equipment in order to clarify how modems select channels. One question we were seeking to answer was whether devices had a preference for a specific channel vice a channel range.

In this experiment, the base stations were assigned channel values within low, mid-low, mid-high, and high ranges corresponding to the Experiment 2 channels of 1, 42, 83, and 124. There was some concern that channel selections of 1 and 124, in particular, would be influenced by these channels being on the edge of the frequency band. Some other wireless protocols, for instance, assign edge channels to control functions instead of communications.

We modified the *randChannel.py* program in Appendix F.C to randomly select a value from the channel configuration ranges shown in Table 15 and then randomly assign which base station gets each value. The channel assignments and the uniform distribution check for this experiment are displayed in Table 16.

Table 15. Experiment 2.2 Configurations

Experiment 2.2: Channel Test Configurations	
Band Configuration	900 MHz
Channel Configuration	Randomize Channel Assignment, Every Round from: Low (L): [5-15] Med-Low (ML): [40-50] Med-High (MH): [75-85] High (H): [110-120]
Round 1	Configuration Mode (Band, Channel), Capture Mode, Template SIM
Rounds 2–10	Configuration Mode (Channel), Capture Mode, Template SIM

Table 16. Experiment 2.2 Channel Assignments and Occurrences

Round	Workstation 1	Workstation 2	Workstation 3	Workstation 4
1	48	118	81	8
2	110	83	12	40
3	75	7	112	41
4	6	117	46	82
5	49	119	13	75
6	10	80	113	43
7	120	15	50	78
8	111	42	75	13
9	78	43	9	116
10	11	79	47	120
Range	**Number of Occurrences per Workstation**			
L	3	2	3	2
ML	2	2	3	3
MH	2	3	2	3
H	3	3	2	2

The results in Table 17 show that the modems again selected workstation three 76% of the time during the first five rounds. Since we noticed this before running the next five rounds, we relocated the SDRs to establish if the modems would still favor one workstation. Moving the SDRs did not appear to have any effect at the spectrum analyzer.

During the last five rounds, the modems selected workstation four 46% of the time for the remaining rounds. This result could suggest that our spectrum analyzer lacks the granularity to display any minor power differences that still encourage cell selection. Over all ten rounds, the UC20 modems selected workstation three 48% of the time.

Table 17. Experiment 2.2 Number of Workstation Occurrences

Workstation	First 5 Rounds	Last 5 Rounds	Total
1	2	9	11
2	8	8	16
3	38	10	48
4	2	23	25

The channel range occurrences provided in Table 18 show that the low range value was selected the most, 37% of the time. The ML selections were the second most selected value, followed behind by the H value and finally the MH value.

Table 18. Experiment 2.2 Channel Range Occurrences by Round

Range	Number of Occurrences										
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	TOTAL
L	1	8	4	0	10	0	2	6	1	5	37
ML	1	0	0	9	0	6	3	3	2	2	26
MH	5	1	0	1	0	1	3	1	2	0	14
H	3	1	6	0	0	3	2	0	5	3	23

The results in Figure 34 provide more clarity for round selections. For the first five rounds, the L or ML values were assigned to workstation three for three of the five rounds. Even though workstation three was selected the most for the first five rounds, more devices selected workstation three when it was assigned a L or ML value for those five rounds. Similarly, this is also the case for workstation four for the final five rounds.

Round		Workstation 1	Workstation 2	Workstation 3	Workstation 4
1	ARFCN	48 (ML)	118 (H)	81 (MH)	8 (L)
	Device	D8	D4 D5 D9	D1 D2 D3 D6 D10	D7
2	ARFCN	110 (H)	83 (MH)	12 (L)	40 (ML)
	Device	D6	D4	D1 D2 D3 D5 D7 D8 D9 D10	
3	ARFCN	75(MH)	7 (L)	112 (H)	41 (ML)
	Device		D1 D2 D5 D8	D3 D4 D6 D7 D9 D10	
4	ARFCN	6 (L)	117 (H)	46 (ML)	82 (MH)
	Device			D2 D3 D4 D5 D6 D7 D8 D9 D10	D1
5	ARFCN	49 (ML)	119 (H)	13 (L)	75 (MH)
	Device			D1 D2 D3 D4 D5 D6 D7 D8 D9 D10	
6	ARFCN	10 (L)	80 (MH)	113 (H)	43 (ML)
	Device		D1	D2 D3 D6	D4 D5 D7 D8 D9 D10
7	ARFCN	120 (H)	15 (L)	50 (ML)	78 (MH)
	Device	D2 D3	D8 D10	D4 D6 D9	D1 D5 D7
8	ARFCN	111 (H)	42 (ML)	75 (MH)	13 (L)
	Device		D3 D4 D7	D5	D1 D2 D5 D8 D9 D10
9	ARFCN	78 (MH)	43 (ML)	9 (L)	116 (H)
	Device	D1 D3	D7 D9	D2	D4 D5 D6 D8 D10
10	ARFCN	11 (L)	79 (MH)	47 (ML)	120 (H)
	Device	D1 D2 D4 D7 D8		D3 D5	D6 D9 D10
Total		11	16	48	25
L		5	6	19	7
ML		1	5	14	6
MH		2	2	6	4
H		3	3	9	8

Figure 34. Experiment 2.2 Capture Results

Table 19 is a modification of Table 18 with the workstation three selections removed from the first five rounds, and the workstation four selections removed from the last five rounds. When selections are removed for workstation three and four, the selections only slightly favor the L and ML values. The total occurrences are close enough that the results could also suggest that the workstation three SDR power was greater during the first five rounds where it was assigned a L or ML value. Likewise, the power for workstation four SDR could have been greater during the rounds it was assigned a L or ML value.

The total selection values have the same order shown in Table 18 even when the favored workstations are removed. The L value was selected most with the ML value as the second most selected. The number of H selections were third and the number of MH selections were last. These results differ from the expectations we had after Experiment 2. For Experiment 2, the UC20 modems selected the higher channels offered. During this experiment, the UC20 modems selected the lower channels. Based on these results, we cannot establish if UC20 modems perform their channel search from low-to-high or high-to-low.

Table 19. Experiment 2.2 Channel Range Occurrences by Round Modified

Range	Number of Occurrences										
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**TOTAL**
L	1	X3	4	0	X3	0	2	X4	1	5	~~37~~ 13
ML	1	0	0	X3	0	X4	3	3	2	2	~~26~~ 11
MH	X3	1	0	1	0	1	X4	1	2	0	~~14~~ 6
H	3	1	X3	0	0	3	2	0	X4	X4	~~23~~ 9

X3: Workstation 3, removed

X4: Workstation 4, removed

F. SUMMARY OF RESULTS

In this chapter we presented four experiments that examined frequency and channel selections across a range of devices. Since we were not able to clearly identify the frequency and channel preferences, we were unable to perform an experiment that examines the IMSI catching speed using those preferences. A list of the selections is provided in Table 20.

Table 20. Summary of Selections

Experiment	Devices (Band/ Channel Chosen)[a]	Workstation with Most Occurrences	Band/Channel with Most Occurrences
1	SGSII English (1800), SGSII Spanish (900,1800), UC20 Modem (1800,1900)	2	1800 MHz
2	SGSII English (83,124), SGSII Spanish (42 and 124), iPhone 6 (124) UC20 Modem (124,83)	2	124
2.1	None Observed	3	None Observed
2.2	UC20 Modem (L, ML)	3	L

[a]Band/Channel lists the most chosen values in order with most occurrences first. If listed occurrences are equivalent, 'and' will be used in place of ','.

For Experiment 1, workstation two was only assigned the 1800 MHz band so the devices listed with possible preferences could favor 1800 MHz or workstation two. Due to this design issue, it is likely that preferences seen here are not definitive. The Apple devices did not show clear preferences. However, they also chose differently from each other, which could still suggest that models between manufacturers differ in decision-making. The SGSII English predominantly selected the 1800 MHz band compared to the SGSII Spanish devices that primarily selected the 1900 MHz band. Since their results slightly contrast each other, the results could suggest that device preferences of the same model and manufacturer also vary depending on intended use location.

In the remaining experiments we examined the channel preferences within the 900 MHz band. The results of Experiment 2 showed that ARFCN 124 was selected the most and that certain devices chose ARFCN 124 as their first or second choice. However, Table 20 shows workstation two was also selected more than the other workstations. Workstation two was assigned ARFCN 124 for four of the ten rounds which is likely the reason for it being selected the most. The SGSII English and Spanish devices as well as the Apple devices again varied their selections which supports the idea that model and manufacturing location could affect decision-making. The UC20 modems had clearer selections than any of the devices during both experiments and one device always differed from the other two. We used the UC20 modems for the remainder of the experiments to focus on one device that displayed clearer selections during Experiments 1 and 2.

We designed Experiment 2.1 to see if the modems would choose the highest randomized value the base stations provided. The UC20 modems selected the highest channel, ARFCN 124, during Experiment 2. The only clear selection we noticed was that the modems favored workstation three instead of workstation two, as in Experiment 2. This result suggests that workstation three was likely the most powerful base station for most of the rounds during the experiment. However, this result also could mean that the UC20s could prefer ARFCN 124 or near ARFCN 124 instead of the highest channel offered.

Experiment 2.2 was designed to examine the range of channels which were categorized into L, ML, MH, and H. This experiment resulted in two possible preferences much like in Experiment 1, even though we randomized the base station assignments. The UC20 modems selected the L and ML values more over the MH and H values. However, the L values were selected more than the ML values. Additionally, workstation three was also selected more than the other workstations. Workstation three was also assigned the L value and ML value three times each during the ten rounds which could explain why it had been selected the most. The final two channel tests contradicted the ARFCN 124 selection from Experiment 2. Therefore, we cannot definitely determine if devices select channels from high-to-low or low-to-high. Furthermore, since the results of Experiment 2.1 showed favoritism for a workstation we cannot determine if devices randomly or pseudo-randomly make channel selections.

Unfortunately, due to delays and restraints resulting from the COVID-19 pandemic, we were unable to refine experiment design quickly enough to adapt to the observed power sensitivities and experiment design flaws. Under other circumstances we would have treated these experiments as “learning experiences” and quickly iterated improvements. As a result, we were unable to conclusively determine if devices consistently favored workstations or a specific frequency band and channel. Even for devices that had clear selections throughout the experiments, we lacked the equipment and time to determine the extent to which power imbalances affected device decision making.

V. CONCLUSIONS

In this thesis, we examined the possibility that IMSI catching attackers can create profiles based on their targeted devices. If target profiles can be created based on the manufacturer or model of a cellular device, attackers could gain an advantage over the legitimate base stations of a commercial network in order to trap a target.

We examined this possibility by first determining the radio resources required for connecting to a GSM network. We also identified the decision-making factors cellular devices use to select cells on a GSM network. We then created an IMSI catcher and designed a testing environment for experimentation. Using our IMSI catchers and testing environment, we performed a set of experiments to examine frequency and channel preferences for a range of cellular devices.

The frequency and channel tests show distinct selections for some devices. The UC20 modems displayed more obvious selections during the frequency and initial channel test, selecting the upper GSM bands and higher channels. However, the UC20 modems did not provide clear selections for a channel during the final two channel tests. The SGSII English and SGSII Spanish devices rarely had similar results for both the frequency and channel tests. This result suggests that the location the phones are manufactured for may have different preferences even though both devices are the same model and manufacturer. Additionally, the iPhone 3 and iPhone 6 devices had different selection results which indicates decision-making could also differ between models even if they are from the same manufacturer.

Based on the experimental data, we could not conclusively differentiate if the cellular devices favored certain selections or the base station that provided those selections. This result implies that the power fluctuations emitted from the SDRs may have influenced device selections. We assess this due to the favored workstations in addition to frequency and channel selections seen in all four experiments. After we noticed this result in Experiment 1, we made changes to the setups for the remaining experiments. Even with the modifications, we still observed favoritism towards workstations in addition to certain

frequencies and channels. During the frequency and channel tests, some devices remained inconsistent with both the frequency/channel selections and workstation occurrences. This result could suggest that the power for the favored workstation fluctuated when the devices chose inconsistently. The results could also imply certain devices or manufacturers are differently sensitized for the received power, causing them to sort their power options differently. As well, the manufacturers could prioritize certain frequency bands over power levels when selecting cells. In other words, if priority bands are within an acceptable power level, then a priority band will be selected, even if a cell with better power is available. We were unable to perform more rounds to determine conclusions that could perhaps clarify the results and issues we experienced.

Due to time constraints and lengthy experimental rounds we were limited in the number of rounds that could be performed for each experiment as well as the number of experiments that could be executed. Therefore, we were also unable to conclusively determine the resource preferences that are necessary to examine the speed of IMSI catching, which was our final goal to test for.

A. SIGNIFICANT CONTRIBUTIONS

The preliminary work described in this thesis documents the steps taken to create and setup an enclosed IMSI catcher testing environment. The significant contribution of this thesis work includes a collection of programs performing three major functions in the testing environment. First, the programs are designed to configure multiple base stations, providing a large testing platform to examine IMSI catching. The second function of the programs uses the configured base stations to capture an IMSI and then perform the final function to log the essential base station selection information.

The development and testing stages of the software tools required approximately a year and a half of preparatory work. Most of this time was dedicated to learning the C programming language and implementing the necessary libraries, described in Appendices F and G. With the created programs, we were able to complete one frequency band test and three channel tests. The results from testing are not conclusive but provide a firm foundation for conducting future experimentation. The analysis of the resulting data

suggests a possibility that devices have radio resource preferences. Furthermore, the data confirms the operability of the programs created and testing environment that can be used for future work.

B. FUTURE WORK

Based on the results from the experiments in Chapter IV, there are several avenues for future work using the test environment provided in this thesis. Since power is one of two contributing factors to the cell selection and reselection processes, any future work will first require a way to eliminate the power fluctuations observed during experimentation. We were able to closely equalize the power peaks of the SDRs through the use of LPFs, a Faraday cage, and attenuators. Even so, the power fluctuations were observed to be as large as 10 dB between the lowest and highest peak at times within the test environment. The Faraday cage we used was not lined with material that would absorb electromagnetic radiation. The absorbent material may be helpful if the signals from the SDRs were reflecting within the Faraday cage and causing the fluctuations.

Based on the results from all four experiments, more rounds for the frequency band test and channel band tests are required. The round time is costly, averaging approximately 45 minutes with 15 of those minutes dedicated to the SIM card template procedure. However, more rounds could conclusively determine if the selections exhibited during experimentation are legitimate preferences. Additionally, more rounds could establish if all devices or only a subset of devices have radio resource preferences. Determining frequency band preferences and channel preferences is essential for future testing possibilities.

Since we could not conclusively determine the frequency band and channel preferences, we were unable to attempt to answer our final question. The third test requires the band and channel preferences in order to ascertain if frequency and channel preferences decrease the IMSI capture speed. A decrease in IMSI capture speed when using the preferences in comparison to randomized valid configurations could imply that devices can be profiled for targeting by IMSI catchers.

The experiment setup presented in this thesis is designed for testing the cell selection process for a cellular device. However, an attacker is more likely to target devices that are already connected to a cell tower and would have to go through a cell reselection process to connect to a rogue base station. We propose a new experiment that would center testing on the cell reselection process using frequency and channel preferences. This method requires measuring the time it takes for a cellular device to go from a fully connected state, down to zero service, and then finally reselect a base station. The setup provided in this thesis work would require only slight modifications to include a base station outside the Faraday cage acting as the cell tower a test phone is already connected to. The test phone would then be walked over the Faraday cage threshold to start the round time.

APPENDIX A. SIM CARD PYSIM CONFIGURATION

The information in this appendix contains the *pySim* [34] configuration for the SIM cards. For simplicity, all of the SIM cards used during experimentation were configured before the experiments according to Figure A.1.

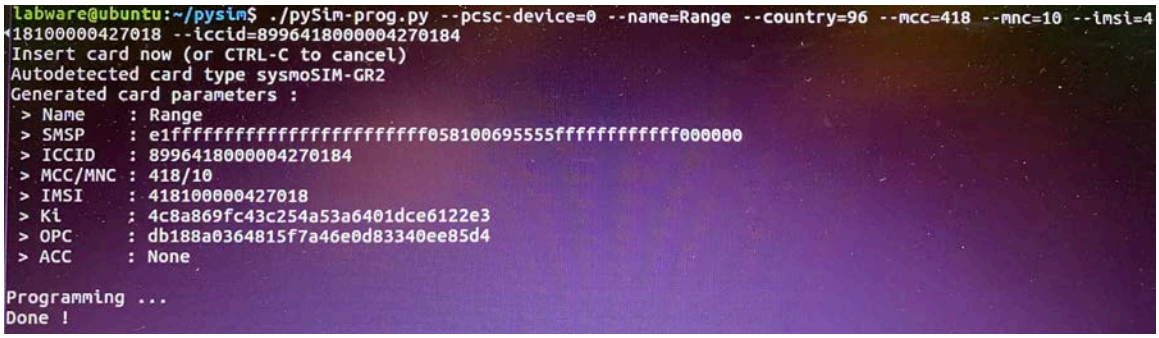

Figure A.1 SIM Card Configurations via pySim.

The options `--name=Range` is for Range Network sim cards and the `--mcc=418 and --mnc=10` set the MCC and MNC to the same Iraq test network that the rogue base stations are configured to. All of the SIM cards were configured to the same IMSI with the 418 10 PLMN (MCC + MNC) and ICCID identifier. Since each phone would be tested one at a time, there was no need for uniqueness on the network during experimentation.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SIM CARD MINICOM CONFIGURATION

This appendix contains the configuration scripts that were run between every round on every SIM card. Also included are instructions for installing and configuring *minicom 2.7*. *Minicom* must be used on a Linux OS installed with the command:

```
sudo apt-get install minicom
```

and must first be configured to the modem connected. In this thesis, a Quectel UC20 modem separate from the testing models is used. Configuring the modem is accomplished using the command:

```
sudo minicom -s
```

where `-s` enters minicom in settings mode. The device port that is connected to the modem is required for entry in the serial port setup. The final step requires saving the configuration as dfl before exiting the settings mode.

The script tempSIM.txt is used to template the SIM for experimentation using the command:

```
sudo minicom -S tempSIM.txt
```

while the seeSIM.txt allows for quickly seeing the template locations with the command:

```
sudo minicom -S seeSIM.txt
```

where the `-S` flag indicates to run a script upon bootup of minicom. Both commands must be run in the same directory the scripts are stored.

A. TEMPSIM.TXT

```
sleep 5
send AT+CRSM=214,28539,0,0,12,\"64F01064F040130062FFFFFF\"
expect “OK”
send AT+CRSM=214,28542,0,0,11,\"FFFFFFFFFFFFFFFFFE0001\"
expect “OK”
send
AT+CRSM=214,28464,0,0,24,\"14F801FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\"
expect “OK”
send AT+CRSM=214,28532,0,0,16,\"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\"
expect “OK”
send AT+QPOWD
exit
```

B. SEESIM.TXT

```
sleep 5
send AT+CRSM=176,28539,0,0,12
expect “OK”
send AT+CRSM=176,28542,0,0,11
expect “OK”
send AT+CRSM=176,28464,0,0,24
expect “OK”
send AT+CRSM=176,28532,0,0,16
expect “OK”
exit
```

C. COMMANDS

The majority of the commands used in this thesis are `AT+CRSM` commands, which are AT commands for restricted SIM access [10]. As a guideline, the AT+CRSM commands listed generally take following format to write to a EF:

AT+CRSM=<Binary Command>,<EF Address in Base10>,0,0,<Byte Length of Location>,

<Hex Information to be Stored at Location>

where the first parameter tells the AT command the action that will take place on the EF. The specifications [10] define the options to be READ BINARY (176), READ RECORD (178), GET RESPONSE (192), UPDATE BINARY (214), UPDATE RECORD (220), STATUS (242), RETRIEVE DATA (203), and SET DATA (219).

The next parameter is the address space of the EF in base 10, or decimal. For example, the file identifier for EF_{BCCH} is 6F74 and is equivalent to 28532 in decimal. The decimal value 28532 would be used in the address space instead of 6F74. The two zeros following the address space value are required by the standard [10]. The byte length indicates the length of space allocated to the information of the address. The final parameter is used when updating a binary and contains the information in hexadecimal format for storage [10]. The final parameter must follow the standards for proper formatting as some locations require nibble (half byte) swaps for each byte [12]. For example, formatting the PLMN `418 10` for $EF_{PLMNsel}$ will translate to `14F801` as the first three bytes for the binary.

APPENDIX C. NTP CONFIGURATION

This appendix contains the ntp.conf files used for the NTP configuration, found at /etc/ntp.conf. The ntp.conf files presented are formatted for use on a Linux OS computer, with A designed for the primary computer acting as the NTP server and B designed for the secondary workstations as the client. Having the correct time isn't necessarily of concern, but instead ensuring that all of the workstations had the same time is the problem to resolve. This is required because we want the IMSI capture time which is calculated by subtracting the capture time stamp from the start time stamp. The most accurate clock time does not matter for the delta time calculation as long as the workstation clocks are the same.

A. NTP.CONF—PRIMARY

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Specify one or more NTP servers.
#server time.nps.edu
server 127.127.1.0 minpoll 4 maxpoll 4
fudge 127.127.1.0

# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011–02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
pool 0.us.pool.ntp.org iburst
pool 1.us.pool.ntp.org iburst
pool 2.us.pool.ntp.org iburst

# Use Ubuntu's ntp server as a fallback.
#pool ntp.ubuntu.com

# Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for
# details. The web page #<http://support.ntp.org/bin/view/Support/
AccessRestrictions>
# might also be helpful.
#
# Note that "restrict" applies to both servers and clients, so a configuration
# that might be intended to block requests from certain clients could also end
# up blocking replies from your own upstream servers.

# By default, exchange time with everybody, but don't allow configuration.
```

```
restrict -4 default kod notrap nomodify nopeer
restrict -6 default kod notrap nomodify nopeer

# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1

restrict 192.168.4.0 mask 255.255.255.0
### Use the subnet address for the LAN

# Needed for adding pool entries
restrict source notrap nomodify noquery

# Clients from this (example!) subnet have unlimited access, but only if
# cryptographically authenticated.
#restrict 192.168.123.0 mask 255.255.255.0 notrust

# If you want to provide time to your local subnet, change the next line.
# (Again, the address is an example only.)
broadcast 192.168.4.255
### Use the subnet address for the LAN

# If you want to listen to time broadcasts on your local subnet, de-comment the
# next lines. Please do this only if you trust everybody on the network!
#disable auth
#broadcastclient
```

B. NTP.CONF—SECONDARY

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Specify one or more NTP servers.

# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011–02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
#server 0.ubuntu.pool.ntp.org
#server 1.ubuntu.pool.ntp.org
#server 2.ubuntu.pool.ntp.org
#server 3.ubuntu.pool.ntp.org
server 192.168.4.255 prefer iburst

# Use Ubuntu’s ntp server as a fallback.
#server ntp.ubuntu.com

# Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for
#details. The web page #<http://support.ntp.org/bin/view/Support/
AccessRestrictions>
```

```
# might also be helpful.
#
# Note that “restrict” applies to both servers and clients, so a configuration
# that might be intended to block requests from certain clients could also end
# up blocking replies from your own upstream servers.

# By default, exchange time with everybody, but don’t allow configuration.
#IPv4
restrict -4 default kod notrap nopeer
#IPv6
restrict -6 default kod notrap nomodify nopeer noquery


# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1

# Clients from this (example!) subnet have unlimited access, but only if
# cryptographically authenticated.
restrict 192.168.4.0 mask 255.255.255.0 nomodify notrap
### Use the subnet address for the LAN

# If you want to provide time to your local subnet, change the next line.
# (Again, the address is an example only.)
#broadcast 192.168.123.255

# If you want to listen to time broadcasts on your local subnet, de-comment the
# next lines. Please do this only if you trust everybody on the network!
disable auth
broadcastclient
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. NTP TROUBLESHOOTING

The information in this appendix provides efforts in troubleshooting NTP. We experienced many issues when first attempting to synchronize workstations that do not touch the Internet due to large gaps in time between the different system clocks. The correct configuration file in Appendix C in combination with this guide should help troubleshoot some issues we experienced. The *primary.c* and *secondary.c* programs as well offer a third mode to assist known as NTP mode. NTP mode can be run on the primary acting as the NTP server to reboot the NTP service. As well, NTP mode can be run on the secondary workstations to query the server, revealing Figure B.1.

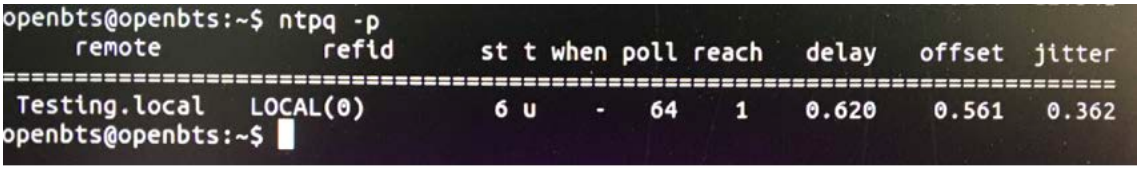

Figure B.1 Picture of ntpq Output on Secondary Workstation.

Alternatively, the same output will be shown using the command

```
ntpq -p
```

where `-p` is used to list the peers taking on the NTP server role on the network [38]. The same output will be shown since NTP mode uses ntpq to query the server. The values listed in the output allow for performance monitoring of the NTP servers on the network. Generally, the host name of the server will be found under the `remote` column but will sometimes list the IP address of the server instead. The `refid` column provides the IP address, with `LOCAL` indicating the local host address [38]. The `st` column indicates the stratum levels 0–16. The 0 represents unspecified sources, 1 is a reference clock which are nationally standardized clock sources, and 2–15 indicate how close the connected server, otherwise known as secondary server, is to the clock source. The 16th stratum is indicative of being unsynchronized to that source [39]. The `t` column determines the type of the server, such as unicast (`u`), local (`l`), broadcast (`b`), etc. The column for `when` displays the

number of seconds that have passed since the host was polled, or essentially the last time a packet was received from the server [38]. The value in the `poll` column is the interval of time, in seconds, the client is set to poll the server [39]. The `reach` column is actually an eight-bit rotating register that displays the octal value of the register at the time of query [39].

Every time a client polls the server, the register shifts to the left, replacing with a zero. When a valid message is received from the server the register shifts with a one filling the slot. Once the server is queried, the register binary values are converted to octal for displaying [38], [39]. The final three columns are all times displayed in milliseconds [40]. The delay column is the roundtrip time while offset indicates the time offset of the server in relation to the host. The final column is the jitter indicating the root mean square offset differences [38].

The delay, offset, and jitters are generally much larger in time upon startup of a new server and new clients. However, we determined that manually setting all workstations dates and times as close together as can be managed before implementing the proper ntp.conf configurations helps cut the delay times down to much lower values in minutes instead of hours. The stratum indicates an issue when the stratum is 16. Stratum 16 can signify the NTP server is not functioning properly or the network may not be properly connected. This mode does not require operating on the primary and secondaries at the same time and can be implemented individually. As a precaution, during the experiments the times were frequently observed for any apparent offsets in time.

APPENDIX E. OPENBTS BOX CONFIGURATION

The code contained in this appendix provides the OpenBTS configuration. Configuring the properties of the individual SDRs occurs through the use of the OpenBTS Command Line Interface (CLI). For our purposes, the SDRs are configured to have the 418 MCC and 10 MNC indicative of a testing network using an Iraq MCC [41]. This is accomplished through the command line with the commands:

```
$./CLI config GSM.Identity.MCC 418
$./CLI config GSM.Identity.MNC 10
```

The rest of the OpenBTS configurations for the USRPs remained at their default value, unless changed by *secondary.c*. Alternatively, using

```
$./CLI config
```

will reveal the list of available configurations.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. PRIMARY CODE

The code contained in this appendix is for the secondary workstation and are used for experimentation. This code is primarily the author's own work. The *primary.c* code requires the installation of several packages before attempting to run, which includes: *libpcap* [42], *tcpdump* [42], *nmap* [43], *fping* [44], and *netcat* [45]. The nmap suite should also include the use of *nping* which is also required for operating *primary.c*. This primary program makes use of a packet sniffer adapted from [46]. The learning modules in Tutorialspoint [47],[48] also assisted in the production of this code. To correctly compile *primary.c* use the command:

```
gcc primary.c -lpcap
```

within the same directory as all of the files in this appendix. Then to run the *primary.c* file is accomplished using the command:

```
sudo ./a.out
```

again, in the same directory as the files of this appendix. The *csvLog.py* program only requires Python [49] and is run from within *primary.c*.

The *randchannel.py* file provides better control over ensuring uniform distribution of the randomization of ARFCNs during experimentation. The code itself does not randomly assign the values nor does it ensure uniform distribution but is a script that can be run separately to provide a set of output to check for uniform distribution before assigning the values during experimentation.

A. PRIMARY.C

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <time.h>
#include <pcap.h>
#include <ctype.h>
#include <unistd.h>

#define MAX 2048

//------------------------------------------------------------------------------------
//                                    primary.c
// This program walks a user through configuring the secondarys and logging
// the useful information for analysis from the secondary workstations.
```

```
// Workstation 1: 192.168.4.22
// Workstation 2: 192.168.4.23
// Workstation 3: 192.168.4.24
// Workstation 4: 192.168.4.25
//-------------------------------------------------------------------------------------

char* freq [] = {"850","900","1800","1900"};
char* hosts [] = {"192.168.4.22","192.168.4.23","192.168.4.24","192.168.4.25"};
int randChannel,trigger;
int trigger;
pcap_t *handle = NULL;
int index850[3],index900[3],index1800[3],index1900[3];
int hostCheck850,hostCheck900,hostCheck1800,hostCheck1900;
int testNum;

//-------------------------------------------------------------------------------------
// Configuration Section
//-------------------------------------------------------------------------------------

//---------------------------------------------------------------------|
// Randomly generates ARFCNs according to Band
//---------------------------------------------------------------------|
int randAssign(int min, int max, char band []){
 int randValue = (max-min)/4;
 int channel [130];
 int index [3];
 srand(time(0));
 int random,value;
 int n = 4;
 for (int z = 0; z < 4; z++){
     int same = 1;
     while (same == 1){
         channel [z] = rand()%randValue+(min+(randValue*z+1));
         if (strcmp(band,"1900") == 0){
             if ((( z > 0 ) && ((channel [z]-channel [z-1]) < 10))){
                 same = 1;
             }
             else if (index1800[z] != 0){
                 for (int x = 0;x<4;x++){
                     if ((channel [z] == index1800[x]) || (abs(channel [z]-index1800[x]))
< 10){
                         same = 1;
                         break;
                     }
                     else{
                         same = 0;
                     }
                 }
             }
             else{
                 same = 0;
             }
         }
         else if(strcmp(band,"1800") == 0){
             if ((( z > 0 ) && ((channel [z]-channel [z-1]) < 10))){
                 same = 1;
             }
             else if (index1900[z] != 0){
                 for (int x = 0;x<4;x++){
                     if ((channel [z] == index1900[x]) || (abs(channel [z]-index1900[x]))
< 10){
                         same = 1;
                         break;
                     }
                     else{
                         same = 0;
                     }
                 }
             }
```

```
            else{
                same = 0;
            }
        }
        else{
            if (( z > 0 ) && ((channel [z]-channel [z-1]) < 10)){
                same = 1;
            }
            else{
                same = 0;
            }
        }
    }
 }
 for (int i = 0;i<4;i++){
     n--;
     if (n == 0){
         random  = 0;
     }
     else{
         random = rand()%(n);
     }
     index [i] = channel [random];
     for (int j = random;j<n;j++){
         channel [j] = channel [j+1];
     }
 }
 if (strcmp(band,"1900") == 0){
     for (int i = 0;i<4;i++){
         index1900[i] = index [i];
     }
 }
 else if (strcmp(band,"1800") == 0){
     for (int i = 0;i<4;i++){
         index1800[i] = index [i];
     }
 }
 else if (strcmp(band,"850") == 0){
     for (int i = 0;i<4;i++){
         index850[i] = index [i];
     }
 }
 else if (strcmp(band,"900") == 0){
     for (int i = 0;i<4;i++){
       if (index [i] == 0){
              index900[i] = 1;
       }
       else{
              index900[i] = index [i];
       }
     }
 }
 return 0;
}
//-------------------------------------------------------------------------|
// Assigns ARFCNs according to user input for band
//-------------------------------------------------------------------------|
int assignARFCN(char band [], char ipAddress []){
 int rchannel;
 if (atoi(band) == 1900){
     // Valid channels for 1900: 512 - 810
     if (randChannel == 0){
         printf("Please enter a valid channel between 512 - 810 for the 1900 band:\n");
         scanf(" %d," &rchannel);
     }
     else if (randChannel != 0){
         if (hostCheck1900 == 0 && randAssign(512,808,"1900") == 0){ // Modified end for
whole number assignment
             rchannel = index1900[0];
```

```
        }
        else{
            rchannel = index1900[hostCheck1900];
        }
        printf("Channel for 1900 [Host: %s] is: %d\n," ipAddress, rchannel);
        hostCheck1900++;
    }
}
else if (atoi(band) == 1800){
    // Valid channels for 1800: 512 - 885
    if (randChannel == 0){
        printf("Please enter a valid channel between 512 - 885 for the 1800 band:\n");
        scanf(" %d," &rchannel);
    }
    else if (randChannel != 0){
        if (hostCheck1800 == 0 && randAssign(512,884,"1800") == 0){ // Modified end for
whole number assignment
            rchannel = index1800[0];
        }
        else{
            rchannel = index1800[hostCheck1800];
        }
        printf("Channel for 1800 [Host: %s] is: %d\n," ipAddress, rchannel);
        hostCheck1800++;
    }
}
else if (atoi(band) == 900){
    // Valid channels for 900: 1 - 124
    if (randChannel == 0){
        printf("Please enter a valid channel between 1 - 124 for the 900 band:\n");
        scanf(" %d," &rchannel);
    }
    else if (randChannel != 0){
        if (hostCheck900 == 0 && randAssign(0,124,"900") == 0){ // Modified end for whole
number assignment
            rchannel = index900[0];
        }
        else{
              rchannel = index900[hostCheck900];
        }
        if (rchannel == 0){
              rchannel = 1;
        }
        printf("Channel for 900 [Host: %s] is: %d\n," ipAddress, rchannel);
        hostCheck900++;
    }
}
else if (atoi(band) == 850){
    // Valid channels for 850: 128 - 251
    if (randChannel == 0){
        printf("Please enter a valid channel between 128 - 251 for the 850 band:\n");
        scanf(" %d," &rchannel);
    }
    else if (randChannel != 0){
        if (hostCheck850 == 0 && randAssign(128,248,"850") == 0){ // Modified end for whole
number assignment
            rchannel = index850[0];
        }
        else{
            rchannel = index850[hostCheck850];
        }
        printf("Channel for 850 [Host: %s] is: %d\n," ipAddress, rchannel);
        hostCheck850++;
    }
}
else{
    printf("Invalid Band Received...\nExiting...\n");
    exit(1);
}
```

```
 return rchannel;
}

//-------------------------------------------------------------------------|
// Determines if all hosts are connected
//-------------------------------------------------------------------------|
int healthCheck(int check){
 FILE *output;
 int size;
 char fping [MAX];
 sprintf(fping,"fping -u %s %s %s %s > output.txt",hosts [0],hosts [1],hosts [2],hosts [3]);
 system(fping);
 output = fopen("./output.txt","r");
 if (output == NULL){
     printf("Error Opening File!\n");
     check = 0;
 }
 else{
     fseek(output,0,SEEK_END);
     size = ftell(output);
     if (size == 0){
         check = 1;
     }
     else{
         check = 0;
     }
 }
 fclose(output);
 return check;
}

//-------------------------------------------------------------------------|
// Sends configurations to hosts
//-------------------------------------------------------------------------|
int antennaAssignment(char band [], char ipAddress [], int count){
 char npingPass [MAX];
 char npingFail [MAX];
 char channelMess [MAX];
 int val;
 int channel;
 if ((strstr(band,freq [0]) != NULL) || strstr(band,freq [1]) != NULL || strstr(band,freq
[2]) != NULL || strstr(band,freq [3]) != NULL){
     channel = assignARFCN(band, ipAddress);
     printf("Sending channel to host..\n");
     sprintf(channelMess,"echo %i | nc %s 444%i",channel,ipAddress,count);
     system(channelMess);
     printf("\nEnsure host received channel, continue? [y = 1/n = 0]: ");
     if (scanf(" %i," &val) != 1){
         printf("Exiting...\n\n");
         exit(1);
     }
     if (count < 4){
         printf("Sending Message to host...\n\n");
         sprintf(npingPass,"nping --udp -p 3333 -g 3333 %s -c 1 -H -N --quiet --data-string
'%s'",ipAddress,band);
         system(npingPass);

     }
     else{
         printf("Sending Message to host...\n\n");
         sprintf(npingFail,"nping --udp -p 3333 -g 3333 %s -c 1 -H -N --quiet --data-string
'%s'",ipAddress,band);
         system(npingFail);
         printf("-------------------------------------------------------\n");
         printf("Ensure hosts are setup correctly on Secondary's\n");

     }
 }
 else{
```

```
     printf("Attempted to assign invalid band...exiting...\n");
     exit(1);
 }
 return 0;
}
int ntpSetup(){
 printf("-------------------------------------------------------\n");
 printf("\nForcing restart of NTP Server...\n");
 system("service ntpd restart");
 system("ntpq -p");
 return 0;
}

//---------------------------------------------------------------------|
// Time stamps data
//---------------------------------------------------------------------|
int timeStamp(int tf,int testNum){
 char csvStamp [MAX];
 sprintf(csvStamp,"python ./csvLog.py %i %i," tf, testNum);
 system(csvStamp);
 return 0;
}

//-----------------------------------------------------------------------------------------
// Base Sniffer Section
//-----------------------------------------------------------------------------------------
void packet(u_char *arg, const struct pcap_pkthdr* hdr, const u_char * packet){
 int i=0;
 char payload [MAX];
 if (trigger != 1){
     if (hdr->len < 135){
         for (i=0; i<hdr->len; i++){
             if (i > 41 && i < 43){
                 strcat(payload, (char *)&packet [i]);
             }
         }
     }
     if ((strstr(payload, hosts [0]) != NULL) || (strstr(payload, hosts [1]) != NULL) ||
(strstr(payload, hosts [2]) != NULL) || (strstr(payload, hosts [3]) != NULL)) {
         printf("\nPayload Received: %s\n\n," payload);
         pcap_breakloop(handle);
     }
     strcpy(payload, "");

 }
}

int packetSniffer(){
 char errorBuff [PCAP_ERRBUF_SIZE], *device;
 device = "enp0s25"; // ----->If necessary, change to whats needed here (device to sniff)

 printf("-------------------------------------------------------\n");
 printf("Setting up packet sniffer...\n");
 handle = pcap_open_live(device, MAX, 1,  512, errorBuff);
 pcap_loop(handle, -1, packet, NULL);

 return 0;

}
//-----------------------------------------------------------------------------------------
// Main: Checks for mode to begin
//-----------------------------------------------------------------------------------------
int main(int argc, char *argv []){
 if (getuid()){
 printf("WARNING: You are not in root. Please rerun this script in root");
 exit(1);
 }
 int check = 0;
 int mode = 0;
```

```
 char hBandSTR[3];
 while (mode == 0){
     if (!(argc>1)){
         printf("Please enter: '0' for Configuration Mode, '1' for Capture Mode, or '+' to
setup NTP.\n");
         char input = getchar();
         if (strcmp(&input,"0") == 0){
             mode = 2;
         }
         else if(strcmp(&input,"1") == 0){
             mode = 3;
         }
         else if(strcmp(&input,"+") == 0){
             mode = 1;
         }
         else{
             printf("Invalid mode...Try Again\n");
         }
     }
     else if(strcmp(argv [1],"+") == 0){
         mode = 1;
     }
     else if(strcmp(argv [1],"0") == 0){
         mode = 2;
     }
     else if(strcmp(argv [1], "1") == 0){
         mode = 3;
     }
     else{
         break;
     }
 }

 switch(mode){
     case 1:{
         if (ntpSetup() == 0){
             printf("\nNTP   Commands   ran   successfully...check   to   ensure   correct
setup...\n");
             timeStamp(0,00000);
         }
         break;
     }
     case 2:{
         trigger = 0;
         printf("-----------------------------------------------------\n");
         printf("ENSURE  SECONDARY'S  ARE  RUNNING  FIRST!  \nIf  not,  quit  and  start  the
secondarys' scripts in Configuration Mode.\n\n");

         while (healthCheck(check) != 1){
             printf("Recheck Computers\n\n");
             printf("Delete output.txt file then press Enter if you would like to retry
health check...\n");
             getchar();
         }
         printf("\nHealth Check Successful!\n");
         printf("\nWould you like to randomize ARFCN Channel assignment? [y = 1/n = 0]: ");
           if ((scanf(" %i",&randChannel) == 1) && (randChannel == 1 || randChannel == 0)){
                 printf("randChannel = %i\n",randChannel);
                 for (int i = 0; i < 4; i++){
                         printf("Please Enter Host %i's Band: \n",i+1);
                         scanf("%s",hBandSTR);
                         antennaAssignment(hBandSTR,hosts [i],i+1);
                         strcpy(hBandSTR,"");
                         packetSniffer();
                    }
                 printf("\nConfiguration of hosts complete!\n");
                 }
          }
     case 3:{
```

```
        char c;
        trigger = 1;
        printf("-----------------------------------------------------\n");
        printf("Enter the test number of the phone here,\n");
        scanf("%i",&testNum);
        printf("Test Phone Number %i\n\n",testNum);


        printf("\nCheck each antenna is configured correctly, if not press q.\n");
        printf("\n1. Run seconday's scripts in Capture Mode. \n");
        printf("2. Check secondary's NTP status', if st==16 press q to quit and try
again.\n");
        printf("3. If all are setup correctly, PRESS ENTER ON SECONDARY'S! \n4. Press Enter
here to begin experiment.\n");
        getchar();
        c = getchar();
        if (strstr(&c,"q")){
            printf("\nExited IMSI experiment.\n");
            exit(1);
        }
        else{
            timeStamp(1,00000);
            printf("\nBeginning Experiment...\n");
            if (system("nc -l 4488 > IMSI.txt") != -1){
                timeStamp(2,testNum);
                char nping [MAX];
                sprintf(nping,"nping --udp -p 3333 -g 3333 %s %s %s %s -c 1 -H -N --quiet
--data-string 'STOP'",hosts [0],hosts [1],hosts [2],hosts [3]);
                system(nping);
                printf("\nSuccessful IMSI experiment!\a\n");
                printf("-----------------------------------------------------\n");
            }
            else{
                printf("\nIMSI Experiment Failed\n");
                printf("-----------------------------------------------------\n");
                exit(1);
            }
        }
    }

 pcap_close(handle);
 return 0;

}
}
```

B. CSVLOG.PY

```
import csv
import sys
from datetime import datetime
# For Primary

#-------------------------------------------------
#              csvLog.py
# This program takes in the necessary data and
# formats it into a .csv file.
#-------------------------------------------------
def main():
       csv = open('imsiLog.csv','a')
       fmt = "%H:%M:%S.%f"
       #-----------------------------------------------
       # Creates the start template for logging
       if (sys.argv [1] == '1'):
              ctime = datetime.now()
              ftime = ctime.strftime(fmt)
              row1 = "START,---,---,---,---,"+ftime+"\n"
              csv.write(row1)
```

```
                time = open('primeTime.txt','w')
                time.write(ftime)
                time.close()
        #-------------------------------------------
        # Creates the header names for the log
        elif (sys.argv [1] == '0'):
                row0 = "Time,ipAddress,Band,TestNumber,Channel,DeltaTime\n"
                csv.write(row0)
        #-------------------------------------------
        # Stamps to the logging and file for matlab
        elif (sys.argv [1] == '2'):
                fileIMSI = open("IMSI.txt","r")
                fileTIME = open("primeTime.txt","r")
                fileMATLAB = open('imsiMATLAB.csv','a')
                readIMSI = fileIMSI.readline()
                readTIME = fileTIME.readline()
                delta        =        datetime.strptime(str(readIMSI)[:14],fmt)        -
datetime.strptime(readTIME,fmt)
                deltaT = str(delta)
                print("\n\nDelta Time: "+deltaT)
                testNum = str(sys.argv [2])
                row = readIMSI.rstrip('\n')+testNum+','+deltaT+"\n"
                rowMATLAB = readIMSI[29:].rstrip('\n')+testNum+','+deltaT+"\n"
                print("Capture Information:\n"+row)
                csv.write(row)
                fileMATLAB.write(rowMATLAB)
                fileMATLAB.close()
                fileIMSI.close()
                fileTIME.close()
        csv.close()
main()
```

C. RANDCHANNEL.PY

```
import random

#-------------------------------------------
#              randChannel.py
# This program was designed to provide
# more control for creating random ARFCNs.
# Check for uniform distribution.
#-------------------------------------------

#----------------------------
# Randomziation for EP2
#----------------------------

channels = [1,42,83,124]
for i in range(10):
        sampled = random.sample(channels,4)
        print(sampled)

#----------------------------
# Randomziation for EP2.2
#----------------------------
chan1 = [5,6,7,8,9,10,11,12,13,14,15]
chan42 = [40,41,42,43,44,45,46,47,48,49,50]
chan83 = [75,76,77,78,79,80,81,82,83,84,85]
chan124 = [110,111,112,113,114,115,116,117,118,119,120]
print('\n')
for j in range(10):
        sampled1 = random.sample(chan1,1)
        sampled42 = random.sample(chan42,1)
        sampled83 = random.sample(chan83,1)
        sampled124 = random.sample(chan124,1)
        chanT = sampled1 + sampled42 + sampled83 + sampled124
        sampleT = random.sample(chanT,4)
        print(sampleT)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. SECONDARY CODE

The code within this appendix is for the secondary workstations and used for experimentation. The experiments performed in this thesis use four secondary workstations labeled as secondary<#>.c. The <#> is replaced with the numbered value for the workstation it is on. For example, this code on workstation one would bear the name *secondary1.c*. Implementing the secondary<#>.c code also requires installation of several of the packages mentioned in Appendix F. This includes: *libpcap* [42], *tcpdump* [42] and *netcat* [45]. This secondary program also relies on a packet sniffer adapted from [46] but is primarily the author's own work and was also produced with the assistance of the many learning modules in Tutorialspoint [47],[48]. Successfully compiling *secondary.c* requires the command:

```
gcc secondary<#>.c -lpcap
```

and is run within the same directory as all of the files in this appendix. Then to run the *secondary.c* file, the command:

```
sudo ./a.out
```

is used, again, within the same directory as the files of this appendix. The *testTime.py* program only requires python [49] and will only be called to run from within secondary<#>.c.

A. SECONDARY<#>.C

```
#include <pcap.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <unistd.h>

#define MAX 1000

const char* ipAddr [] = {“192.168.4.22”}; // Workstation 1 ------- Change Here for other
IPaddresses
// Workstation 2: “192.168.4.23”
// Workstation 3: “192.168.4.24”
// Workstation 4: “192.168.4.25”
pcap_t *handle = NULL;
int trigger = 0;
int ARFCN = 0;
int band = 0;
```

```
//-------------------------------------------------------------------------------------------
//                                  secondary#.c
// This program is walked through by the primary. Only starting this script is necessary.
// In capture mode it will listen for STOP or IMSIs and relay data back to primary
//-------------------------------------------------------------------------------------------

//-------------------------------------------------------------------------------------------
// Configuration Section
//-------------------------------------------------------------------------------------------
//-----------------------------------------------------|
// Assigns ARFCN based on requirements from Primary
//-----------------------------------------------------|
int assignARFCN(){
 FILE *fp;
 char str [5];
 int channel;
 fp = fopen("channel.txt," "r");
 if(fp == NULL) {
   perror("Error opening file\nExiting...\n");
   exit(-1);
 }
 if( fgets (str, 60, fp) != NULL ) {
 if (atoi(str) == 0){
     printf("\nChannel Given: %i\n," 1);
         channel = 1;
 }
 else{
         printf("\nChannel Given: %s\n," str);
         channel = atoi(str);
 }
 }
 fclose(fp);
 return channel;

}
//-----------------------------------------------------|
// Configures openBTS
//-----------------------------------------------------|
int configWindow(){
 char mainBand [MAX]; // Changes band in main configuration file
 char pyBand [MAX]; // Ensures antenna band is configured correctly via CLI
 char mainChannel [MAX]; // Changes channel in main configuration file
 char pyChannel [MAX]; // Ensures antenna channel is configured correctly via CLI

 sprintf(mainBand,    "sqlite3    ../../../etc/OpenBTS/OpenBTS.db    'UPDATE    CONFIG    SET
VALUESTRING=\"%i\" WHERE KEYSTRING=\"GSM.Radio.Band\"'",band);
 sprintf(mainChannel,   "sqlite3   ../../../etc/OpenBTS/OpenBTS.db   'UPDATE   CONFIG   SET
VALUESTRING=\"%i\" WHERE KEYSTRING=\"GSM.Radio.C0\"'",ARFCN);
 system(mainBand);
 system(mainChannel);

 // Ensures, files are configured
 sprintf(pyBand, "../../../OpenBTS/OpenBTSCLI -c config GSM.Radio.Band %i," band);
 sprintf(pyChannel,"../../../OpenBTS/OpenBTSCLI -c config GSM.Radio.C0 %i",ARFCN);
 system(pyBand);
 system(pyChannel);

 printf("\nConfiguration complete, restarting OpenBTS...\n");
 system("../../../OpenBTS/OpenBTSCLI -c restart");

 return 0;
}
//-----------------------------------------------------|
// Displays configuration messages for confirmation
//-----------------------------------------------------|
int configurationMessages(int bandMain){
 pcap_breakloop(handle);
 FILE *bandText;
 band = bandMain;
```

```
 bandText = fopen("band.txt","w");
 char bandString [6];
 sprintf(bandString,"%i",band);
 fputs(bandString,bandText);
 fclose(bandText);
 char reply [MAX];
 printf("\nMessage Received Matches: %i\n," band);
 printf("\nConfiguring Antenna For %i Band and Channel %i\n...\n," band, ARFCN);
 configWindow();
 return 0;
}

//-------------------------------------------------------------------------------------
// Base Sniffer Section
//-------------------------------------------------------------------------------------
void packetAnalysis(u_char *arg, const struct pcap_pkthdr* header, const u_char * packet){
 int i=0;
 char payload [129];
 switch(trigger){
     // Checks if there are any configuration intructions from the Primary
     case 1: {
         if (header->len < 135){
             for (i=0; i<header->len; i++){
                 if (i > 41){
                     if ( isprint(packet [i]) ){
                         strcat(payload, (char *)&packet [i]);
                     }
                 }
                 else{
                     strcat(payload, "");
                 }
             }
         }
         if (strstr(payload, "850") != NULL) {
             configurationMessages(850);
         }
         else if (strstr(payload, "900") != NULL && strstr(payload, "1900") == NULL) {
             configurationMessages(900);
         }
         else if (strstr(payload, "1800") != NULL) {
             configurationMessages(1800);
         }
         else if (strstr(payload, "1900") != NULL) {
             configurationMessages(1900);
         }
         strcpy(payload, "");
         break;
         }
 //-------------------------------------------------------------------------------------
 // Capture IMSI Section
 //-------------------------------------------------------------------------------------
     case 2: {
         char str [100];
         const char check [] = "01 3f 49 05 08";
         char stopCheck [2048];
         if (header->len < 90){
             for (i=0; i<header->len; i++){
                 if (i > 40){
                     sprintf(str,"%02x ," (unsigned char) packet [i]);
                     strcat(payload,str);
                     strcpy(str,"");
                     if ( isprint(packet [i]) ){
                     strcat(stopCheck, (char *)&packet [i]);
                     }
                 }
             }
         }
         if (strstr(stopCheck,"STOP")!=NULL){
             printf("Received STOP, experiment ending. Exiting....\n\n");
```

```
            pcap_breakloop(handle);
            exit(1);
        }
        else if (strstr(payload,check)!=NULL){
            printf("\nIMSI FOUND!\nProcessing...\n");
            pcap_breakloop(handle);
            // Need this section only if requirement to see IMSI------------
            int count = 1;
            char IMSItemp [24],IMSI[15],message [200];
            int size = strlen(payload);
            int x,z;
            for (x=0;x<24;x++){
            IMSItemp [23-x] = payload [size-7-x];
            }
            for (z = 0; count < 15;z+=3){
            if (z == 0){
                IMSI[0] = IMSItemp [0];
            }
            else{
                IMSI[count] = IMSItemp [z+1];
                IMSI[count+1] = IMSItemp [z];
                count+=2;
            }
            }
            printf(" IMSI is: %s.\nTime stamping and sending to primary...\n",IMSI);
            //-------------------------------------------------------

            // Sends configuration to time stamper
            sprintf(message,"python ./testTime.py %i %i %s",band,ARFCN,ipAddr [0]);
            system(message);

        }
        else{
                strcpy(payload,"");
                strcpy(stopCheck,"");
        }
    }
 }
}

int packetSniffer(){
 char errorBuff [PCAP_ERRBUF_SIZE], *interface;
 printf("-------------------------------------------------\n");
 if (trigger == 1){
     interface = "eth0"; //----If necessary, change to whats needed here (device to sniff)
 }
 else if (trigger == 2){
     interface  = "any";
 }
 printf("Setting up packet sniffer...\n");
 handle = pcap_open_live(interface, 2048, 1,  512, errorBuff);
 pcap_loop(handle, -1, packetAnalysis, NULL);
 return 0;


}


//-------------------------------------------------------------------------------------
// Main: Checks for mode to begin
//-------------------------------------------------------------------------------------
int main(int argc, char *argv [] ){
 if (getuid()){
 printf("WARNING: You are not in root. Please rerun this script in root");
 exit(1);
 }
 int mode = 0;
 printf("-------------------------------------------------\n");
 printf("Your address is set to: %s\n",ipAddr [0]);
```

```
 while (mode == 0){
     // Checks if an argument was entered. If not, it will ask which mode to select
     if (!(argc>1)){
         char input;
         printf("Please enter a '0' for Configuration Mode, '1' for Capture Mode, or '+' to
check NTP\n");
         input = getchar();
         if (strcmp(&input,"0") == 0){
             mode = 1;
             system("nc  -l  4442  >  channel.txt");  //-----------------------------------
Change Here (nc port) for other IP Address
             ARFCN = assignARFCN();
         }
         else if(strcmp(&input,"1") == 0){
             mode = 2;
             ARFCN = assignARFCN();
         }
     else if(strcmp(&input,"+") == 0){
             mode = 3;
             printf("Checking NTP...\n");
         }
         else{
             printf("\nInvalid Entry, please try again...\n");
         }
     }
     // If argument was given, compares and sets it to correct mode
     else if(strcmp(argv [1],"0") == 0){
         mode = 1;
         system("nc -l 4442 > channel.txt"); //- Change nc port for other IP Address
               // 4443, 4444, 4445
         ARFCN = assignARFCN();
     }
     else if(strcmp(argv [1], "1") == 0){
         mode = 2;
         ARFCN = assignARFCN();
     }
     else if(strcmp(argv [1], "+") == 0){
         mode = 3;
         printf("Checking NTP...\n");
     }
     else{
         break;
     }
 }
 switch(mode){
     // Configuration Mode
     case 1:{
         trigger = 1;
         if (packetSniffer() != 0){
             printf("\nUnsuccessful Configuration");
         printf("-------------------------------------------------\n");
         }
         else{
             printf("\nConfiguration complete!\n");
         printf("-------------------------------------------------\n");
         }
     }
     // Capture Mode
     case 2:{
         trigger = 2;
         if (packetSniffer() != 0){
             printf("\nUnsuccessful IMSI experiment\n");
             printf("-------------------------------------------------\n");
         }
         else{
             printf("\nIMSI Experiment Complete!\n");
             printf("-------------------------------------------------\n");
         }
         break;
```

```
    }
    // NTP Mode
    case 3: {
        system("ntpq -p");
        printf("\nEnsure ntp server is up and st is low\n");
    break;
    }

 }
 pcap_close(handle);
 return 0;

}
```

B. TESTTIME.PY

```
import sys
import os
from datetime import datetime
# For Secondary

#-------------------------------------------------
#                          testTime.py
# This program takes in the necessary data and
# formats it to send to primary
#-------------------------------------------------
def main():
       # Formats data and sends to Primary
       ctime = datetime.now()
       ftime = ctime.strftime('%H:%M:%S.%f')
       band = sys.argv [1]
       ipAddr = sys.argv [3]
       if band == '0':
              fileBand = open("band.txt","r")
              band = fileBand.readline()
       if len(sys.argv [2]) < 2:
              channel = "00" + sys.argv [2]
       elif len(sys.argv [2]) < 3:
              channel = "0" + sys.argv [2]
       else:
              channel = sys.argv [2]
       cmd = 'printf "%s\n" ' + '"' + ftime +","+ ipAddr + ","+ band +","+ channel
+","+ '" ' + "|" + " nc 192.168.4.28 4488"
       print(cmd)
       os.system(cmd)
main()
```

LIST OF REFERENCES

[1] GSMA, “The mobile economy 2020.” Accessed March 3, 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/

[2] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert, “New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities,” in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks—WiSec ‘19*, Miami, Florida, 2019, pp. 221–231, [Online]. doi: 10.1145/3317549.3319728.

[3] R.-P. Weinmann, “Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks,” presented at the WOOT, 2012, p. 10, [Online]. Available: https://www.usenix.org/system/files/conference/woot12/woot12-final24.pdf

[4] T. B. Retterstøl, “Base station security experiments using USRP,” Norwegian University of Science and Technology, 2015. [Online]. Available: http://hdl.handle.net/11250/2359801

[5] Range Networks. OpenBTS. Accessed April 30, 2020. [Online]. Available: http://openbts.org/get-the-code/

[6] *3rd Generation Partnership Project Multiplexing and Multiple Access on the Radio Path Technical Specification*, 3GPP TS 5.02 (Release 4) 2005. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=255

[7] *3rd Generation Partnership Project Radio Subsystem Link Control Technical Specification*, 3GPP TS 5.08 (Release 1999) 2005. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=262

[8] *3rd Generation Partnership Project Numbering, Addressing, and Identification Technical Specification*, 3GPP TS 23.003 (Release 16) 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729

[9] *3rd Generation Partnership Project Mobile Radio Interface Layer 3 Technical Specification*, 3GPP TS 24.008 (Release 16) 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1015

[10] *3rd Generation Partnership Project AT Command Set for User Equipment (UE) Technical Specification*, 3GPP TS 27.007 (Release 16) 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1515

[11] *3rd Generation Partnership Project GSM/EDGE Radio Resource Control Protocol Technical Specification*, 3GPP TS 44.018 (Release 15) 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2686

[12] *3rd Generation Partnership Project Subscriber Identity Module—Mobile Equipment (SIM-ME) Interface Technical Specification*, 3GPP TS 51.011 (Release 4) 2005. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2793

[13] L. Perkov, A. Klisura, and N. Pavkovic, "Recent advances in GSM insecurities," in *2011 Proceedings of the 34th International Convention MIPRO*, Opatija, Croatia, May 2011, pp. 1502–1506, [Online]. Available: https://ieeexplore.ieee.org/document/5967298

[14] B. Harmat *et al.*, "The Security implications of IMSI catchers," in *Proceedings of the 2015 World Congress in Computer Science, Computer Engineering, and Applied Computing.*, Las Vegas, NV, Jul. 2015, p. 6, [Online]. Available: http://worldcomp-proceedings.com/proc/p2015/SAM7035.pdf

[15] D. Strobel, "IMSI catcher," Seminararbeit Ruhr-Universitat Bochum, 2007. [Online]. Available: https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/imsi_catcher.pdf

[16] A. Mruz, "Mobile network security experiments with USRP," Norwegian University of Science and Technology, 2016. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2410685

[17] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl, "IMSI-catch me if you can: IMSI-catcher-catchers," in *Proceedings of the 30th Annual Computer Security Applications Conference on—ACSAC '14*, New Orleans, Louisiana, 2014, pp. 246–255, doi: 10.1145/2664243.2664272.

[18] J. Eberspächer, H.-J. Vögel, C. Bettstetter, and C. Hartmann, *GSM—Architecture, Protocols and Services*, Third Edition. Chichester, U.K.: John Wiley & Sons Ltd g, 2009.

[19] H. Welte, "Anatomy of contemporary GSM cellphone hardware," 2010. [Online]. Available: ftp://ftp.freecalypso.org/pub/GSM/gsm_phone_anatomy.pdf

[20] Osmocom, “Wiki—Qualcomm Linux modems by Quectel & Co—Open source mobile communications.” accessed March 30, 2020. [Online]. Available: https://osmocom.org/projects/quectel-modems/wiki

[21] *UC20 AT Commands Manual,* V1.0. Quectel, Shanghi, China, May 28, 2013. [Online]. Available: https://www.quectel.com/UploadImage/Downlad/UC20_AT_Commands_Manual_V1.0.pdf

[22] OpenEmbedded, “Openembedded.org.” Accessed April 21, 2020. [Online]. Available: http://www.openembedded.org/wiki/Main_Page

[23] *Series E: Overall Network Operation, Telephone Service, Service Operation and Human Factors*. International Telecommunication Union, 2006, [Online]. Available: https://www.itu.int/rec/T-REC-E.118-200605-I

[24] F. van den Broek, “Catching and understanding GSM-signal,” Radbound University Nijmegen, 2010. [Online]. Available: http://www.cs.ru.nl/~F.vandenBroek/pub/scriptie.pdf

[25] M. Mouly and M.-B. Pautet, *The GSM System for Mobile Communications*. Palaiseau, France: Cell & Sys. Correspondence, 1992.

[26] N. Ibrahim, N. A. Naqbi, F. Iqbal, and O. AlFandi, “SIM card forensics: Digital evidence,” in Annual ADFSL Conf. on Digital Forensics, 2016. [Online]. Available: https://commons.erau.edu/adfsl/2016/thursday/3

[27] Ettus Research, “Ettus research—The leader in software defined radio (SDR),” accessed April 07, 2020. [Online]. Available: https://www.ettus.com/

[28] OpenBTS, “About | OpenBTS.” Accessed April 07, 2020. [Online]. Available: http://openbts.org/about/

[29] *OpenBTS Application Suite User Manual.* Range Networks, Apr. 15, 2014, [Online]. Available: http://openbts.org/site/wp-content/uploads/2014/07/OpenBTS-4.0-Manual.pdf

[30] IplinkME, “IplinkME-Knowledge.” Accessed March 30, 2020. [Online]. Available: http://kb.iplinkme.com/

[31] *Quectel_UC20_UMTS HSPA_Specification,* V1.7 Quectel, accessed April 30, 2020. [Online]. Available: https://www.quectel.com/UploadFile/Product/Quectel_UC20_UMTS%20HSPA_Specification_V1.7.pdf

[32] *Minicom Manual Pages,* Accessed March 30, 2020. [Online]. Available: https://linux.die.net/man/1/minicom

[33] Range Networks, “20 blank SIMs,” Accessed April 30, 2020. [Online]. Available: https://rangenetworks.com/store/20-blank-sims

[34] Osmocom, pySim. Accessed April 30, 2020. [Online]. Available: https://github.com/osmocom/pysim

[35] Network Time Protocol Project, “ntp.org: Home of the Network Time Protocol.” Accessed April 30, 2020. [Online]. Available: http://www.ntp.org/

[36] Wireshark, Wireshark. [Online]. Available: https://www.wireshark.org/#download

[37] IMEI.info, “IMEI CHECK—Free online IMEI number checker | IMEI.info.,” Accessed May 05, 2020. [Online]. Available: https://www.imei.info/apple-sn-check/

[38] *ntpq Manual Pages*, Accessed April 29, 2020. [Online]. Available: https://www.freebsd.org/cgi/man.cgi?query=ntpq&sektion=8&apropos=0&manpath=FreeBSD+12.1-RELEASE+and+Ports

[39] D. L. Mills, *Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space*, 2nd ed. CRC Press, 2017. [Online]. https://doi.org/10.1201/b10282

[40] NTP Project Documentation, “NTP debugging techniques.” Accessed May 02, 2020. [Online]. Available: http://doc.ntp.org/4.2.0/debug.html

[41] Mobile Country Codes (MCC) and Mobile Network Codes (MNC), “Most up to date list of MCC and MNC codes: mobile country codes—mobile network codes,” Accessed March 02, 2020. [Online]. Available: https://www.mcc-mnc.com/

[42] T. T. Group, Tcpdump/Libpcap public repository. Accessed April 30, 2018. [Online]. Available: https://www.tcpdump.org

[43] Nmap, Accessed April 28, 2020. [Online]. Available: https://nmap.org/download.html

[44] D. Schweikert, “fping Homepage,” V.4.2. [Online]. Available: https://fping.org/

[45] The GNU Netcat Project, 2004, Netcat. Accessed April 30, 2018. [Online]. Available: http://netcat.sourceforge.net/download.php

[46] L.M Garcia “Programming with Libpcap—Sniffing the network from our own application,” *Hakin9 IT Security Magazine*, vol. 3, no. 2, pp. 38–46, Feb. 2008, Accessed on April 30, 2020. [Online]. Available: https://web.archive.org/web/20191223043917/http://recursos.aldabaknocking.com/libpcapHakin9LuisMartinGarcia.pdf

[47] Tutorialspoint, “Learn C programming.” [Online]. Available: https://www.tutorialspoint.com/cprogramming/index.htm

[48] Tutorialspoint, “C standard library reference tutorial.” [Online]. Available: https://www.tutorialspoint.com/c_standard_library/index.htm

[49] Python, “Python V2.7.” [Online]. Available: https://www.python.org/downloads/

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California